FORTINET

New Features Guide

SD-WAN with FortiOS, FortiManager, and FortiAnalyzer 7.2.x

FORTINET DOCUMENT LIBRARY
https://docs.fortinet.com

FORTINET VIDEO GUIDE
https://video.fortinet.com

FORTINET BLOG
https://blog.fortinet.com

CUSTOMER SERVICE & SUPPORT
https://support.fortinet.com

FORTINET TRAINING & CERTIFICATION PROGRAM
https://www.fortinet.com/training-certification

NSE INSTITUTE
https://training.fortinet.com

FORTIGUARD CENTER
https://www.fortiguard.com

END USER LICENSE AGREEMENT
https://www.fortinet.com/doc/legal/EULA.pdf

FEEDBACK
Email: techdoc@fortinet.com

April 11, 2022
SD-WAN with FortiOS, FortiManager, and FortiAnalyzer 7.2.x New Features Guide
01-720-792697-20220411

TABLE OF CONTENTS

Change Log

Date	Change Description
2022-04-11	Initial release.

Overview

This guide provides details of new features for SD-WAN introduced in FortiOS 7.2, FortiManager 7.2, and FortiAnalyzer 7.2.

For each feature, the guide provides detailed information on configuration, requirements, and limitations, as applicable. For features introduced in FortiManager or FortiAnalyzer, the short product name is appended to the end of the topic heading, for example FMG or FAZ.

Feature	Details
ADVPN	• Phase 2 selectors and ADVPN shortcut tunnels on page 6 • SD-WAN members' local cost exchange on ADVPN shortcut tunnels on page 6
Provisioning	• SD-WAN overlay templates FMG on page 7 • Metafield support on dynamic objects FMG on page 12 • Model device blueprints FMG on page 14 • Application categories in SD-WAN rules FMG on page 17
Reporting	• SD-WAN chart to include more ADVPN shortcut information FAZ on page 24 • SD-WAN chart for MOS scoring FAZ on page 26
Routing	• SD-WAN segmentation over a single overlay on page 31 • Multiple members per SD-WAN neighbor configuration on page 46 • SD-WAN in large scale deployments on page 52 • Route map rules and BGP routes on page 63 • BGP socket limit increase on page 63 • IKE embryonic limit on page 63
SD-WAN steering	• Allow application category as an option for SD-WAN rule destination on page 65 • Add mean option score calculation and logging in performance SLA health checks on page 68
WAN remediation	• Duplication on-demand when SLAs in the configured service are matched on page 71

ADVPN

7.2.0

Phase 2 selectors and ADVPN shortcut tunnels

Phase 2 selectors can be used to inject IKE routes on the ADVPN shortcut tunnel. When configuration method (`mode-cfg`) is enabled in IPsec phase 1 configuration, enabling `mode-cfg-allow-client-selector` allows custom phase 2 selectors to be configured. By also enabling the addition of a route to the peer destination selector (`add-route`) in the phase 1 configuration, IKE routes based on the phase 2 selectors can be injected. This means that routes do not need to be reflected on the hub to propagate them between spokes, avoiding possible BGP daemon process load issues and improving network scalability in a large-scale ADVPN network.

For details, see SD-WAN in large scale deployments on page 52.

SD-WAN members' local cost exchange on ADVPN shortcut tunnels

SD-WAN members' local cost can be exchanged on the ADVPN shortcut tunnel so that spokes can use the remote cost as tiebreak to select a preferred shortcut. If multiple shortcuts originate from the same member to different members on the same remote spoke, then the remote cost on the shortcuts is used as the tiebreak to decide which shortcut is preferred.

For details, see SD-WAN in large scale deployments on page 52.

Provisioning

7.2.0

SD-WAN overlay templates - FMG

Most SD-WAN deployments require complex overlay configurations for datacenter or cloud connectivity. FortiManager 7.2.0 includes an SD-WAN overlay template with a wizard to automate and simplify the process using Fortinet's recommended IPsec and BGP templates.

This topic includes the following.

For more information, including editing a template and onboarding new SD-WAN branch devices, see the FortiManager Administration Guide.

Prerequisites and network planning

Prerequisites

- Import the FortiGate devices that will make up the hub and branch devices into FortiManager.
- Configure the ISP links and other interfaces on your imported devices.
- Create a device group for your branch devices.

Network planning

- Allocate the overlay network address space. By default, the template uses `10.10.0.0/16`.
- Allocate the loopback IP address space. By default, the template uses `172.16.0.0/16`.
- Select an AS number for BGP for the new SD-WAN overlay region. By default, the template uses `65000`.

Using the SD-WAN overlay template

To use the SD-WAN overlay template:

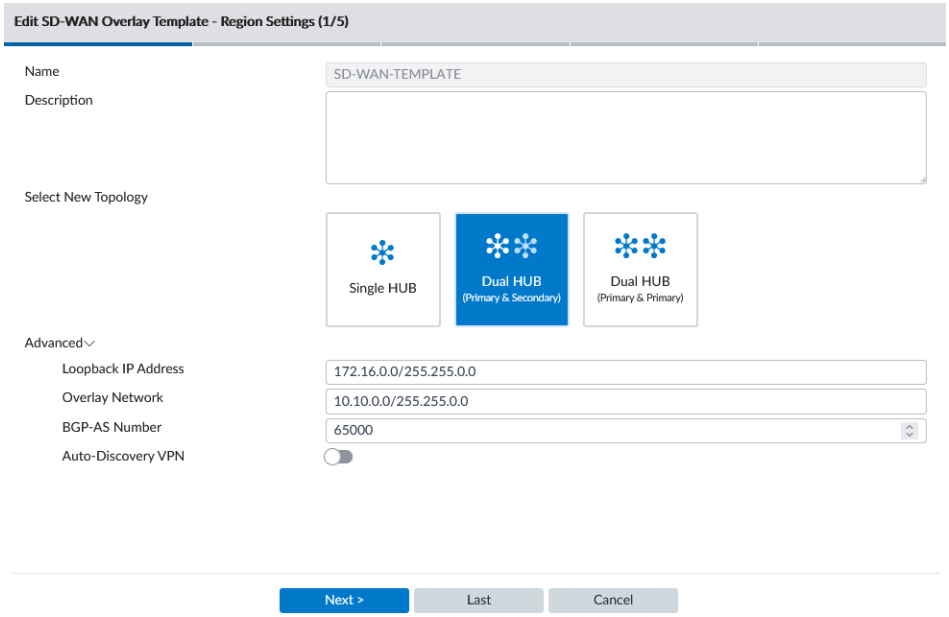
1. Pre-configure your network and SD-WAN devices.
2. Create an SD-WAN overlay template.

3. Assign metadata variables to devices. The branch_id variable is automatically created by the template and each branch device must be assigned a unique value. Additional custom metadata variables can be used if required.
4. Configure the SD-WAN rules to be used in your SD-WAN environment by editing the SD-WAN template.
5. Create the Policy Package for your branch and hub devices.
6. Install the changes to SD-WAN devices using the Install Wizard.
7. (Optional) Edit the SD-WAN overlay template.
8. (Optional) Add new branch devices.

Configuring an SD-WAN overlay template

To create an SD-WAN overlay template:

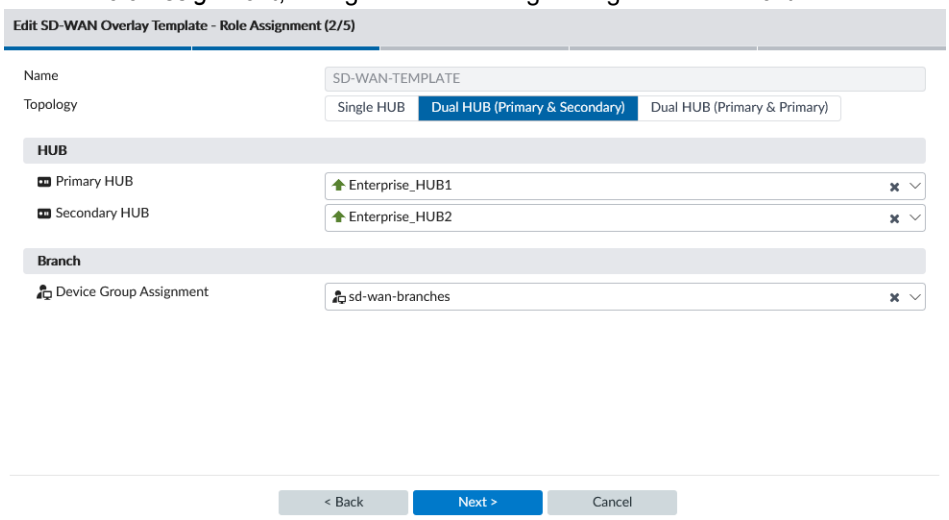
1. Go to *Device Manager* > *Provisioning Templates* > *SD-WAN Overlay Templates*.
2. Click *Create New*.
   The Create New SD-WAN Overlay Template wizard opens.
3. Enter a name and description for the new SD-WAN overlay template, and click *OK*.
4. For the *Region Settings*, select a topology type, and click *Next*.



Select New Topology	Select a topology type based on your environment. Topologies include the following: • Single Hub • Dual Hub (Primary/Secondary) • Dual Hub (Primary/Primary) The options presented in the wizard change based on the topology selected.

		Primary/Secondary and Primary/Primary are the same configuration, with the difference being that in a Primary/Secondary deployment, the Secondary hub is given a higher cost than the Primary. This cost is controlled by the SDWAN rule.
Advanced		Expand to view additional configurable settings. These fields are preconfigured with settings that will work in many situations, but you may need to adjust these to match your own networking environment. They should match the addresses you identified when considering the SD-WAN overlay template prerequisites.
	Loopback IP Address	Optionally, you can configure the loopback IP address. By default, this setting is set to `172.16.0.0/255.255.0.0`.
	Overlay Network	Optionally, you can configure the overlay network. By default, this setting is set to `10.10.0.0/255.255.0.0`.
	BGP-AS Number	Optionally, you can configure the BGP AS number. By default, this setting is set to `65000`.
	Auto-Discovery VPN	Optionally, you can toggle this setting ON to enable Auto Discovery VPN (ADVPN).

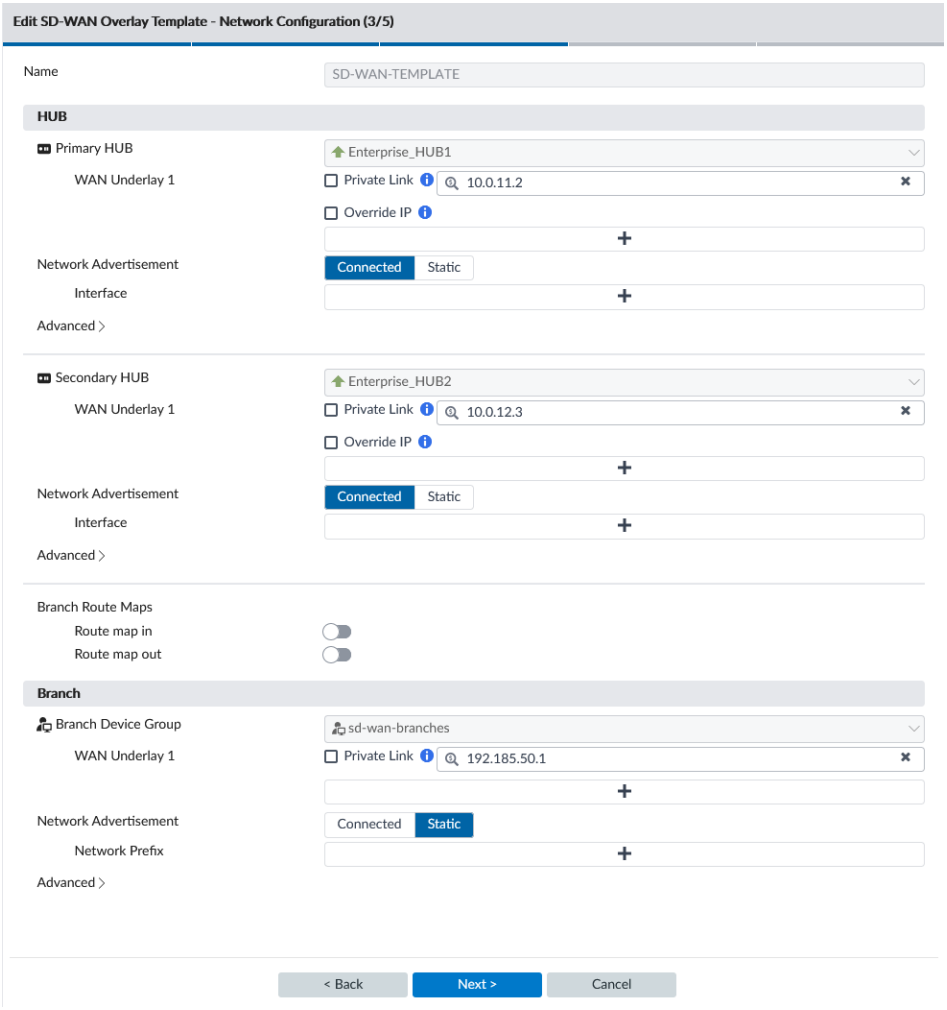
5. For the *Role Assignment*, configure the following settings and click *Next*.



Topology	Optionally, you can change the topology type that you selected on the previous screen.
Hub	Select the SD-WAN hubs. The number of hubs required depend on the topology selected: • *Single Hub*: One standalone hub. • *Dual Hub (Primary & Secondary)*: One primary and one secondary hub. • *Dual Hub (Primary & Primary)*: Two primary hubs.

	Hub devices must be added to SD-WAN with FortiOS, FortiManager, and FortiAnalyzer before creating the SD-WAN overlay template.
Branch	Select the device group containing your SD-WAN branch devices. Devices included in this device group are configured as SD-WAN branch devices as a part of this template. Additional devices can be added to the selected device group later to receive the SD-WAN branch configuration when performing an installation on that device. This simplifies the onboarding of new branch devices.

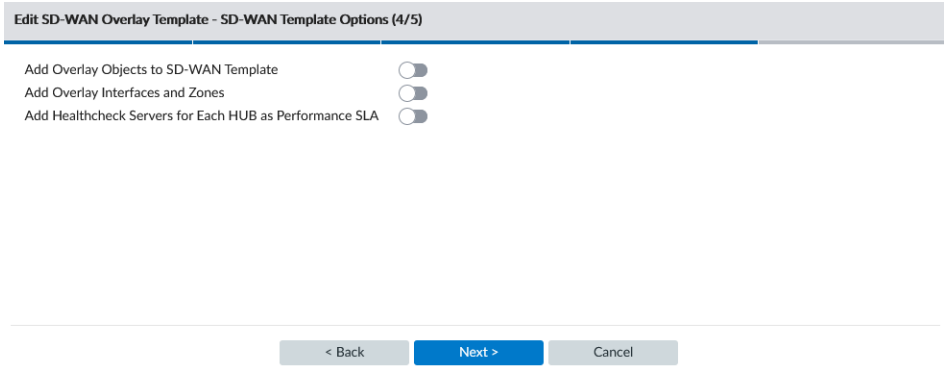
6. For the *Network Configuration*, configure the following settings and click *Next*.



Hub		Configure the network settings for each hub in your configuration. The number and types of hubs present depend on the topology you selected.
	WAN Underlay	Type the interfaces for each WAN underlay. You can add additional WAN underlays by clicking the add icon. For each WAN underlay, you can optionally enable the following settings: • *Private Link*: No overlays will be created on private links.

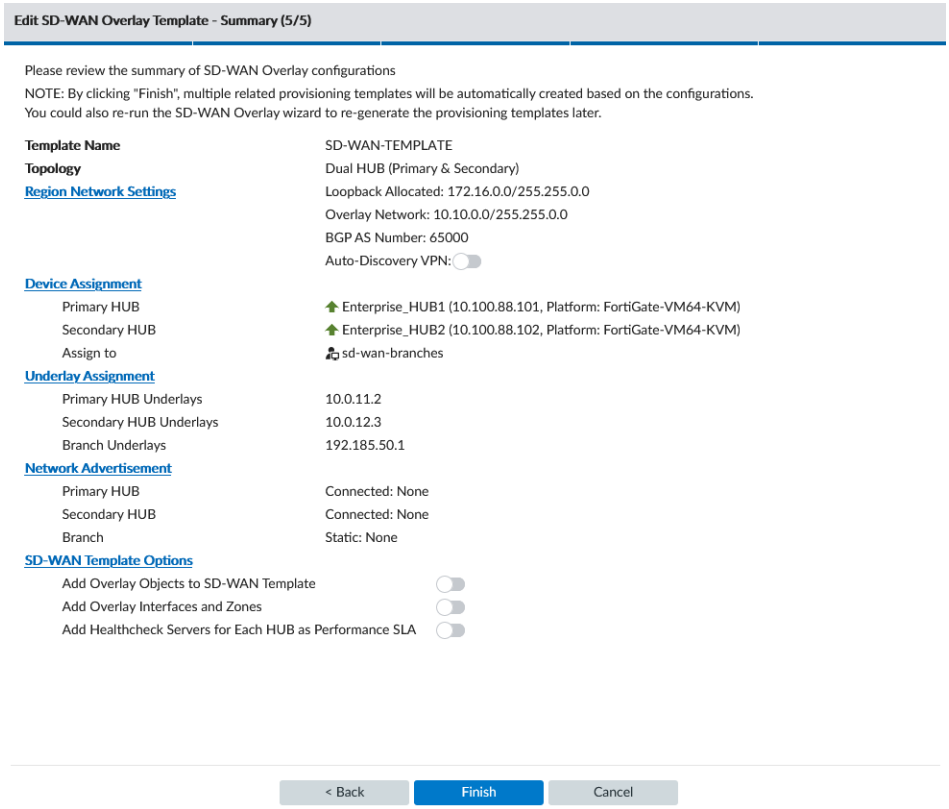
		• *Override IP*: Override the IP address for the WAN underlay with the provided IP address. This option is not available when *Private Link* is enabled.
	Network Advertisement	1. Configure network advertisement for the hub. Network advertisement can be set to one of the following: • *Connected*: Type the network interface to advertise. Additional interfaces can be added by clicking the add icon. • *Static*: Type the network prefix to advertise. Additional network prefixes can be added by clicking the add icon.
	Advanced	Expand to view advanced settings, including configuration of SD-WAN neighbors. Click *Neighbors* > *Create New* to add a new SD-WAN neighbor for the hub.
Branch Route Maps		Optionally, move the toggle to the ON position to enable branch maps, and then select the corresponding route map. You can create a new route map by clicking the add icon.
Branch		Configure the network settings for the branch devices in your configuration.
	WAN Underlay	Type the interfaces for the SD-WAN branch WAN underlay. You can add additional WAN underlays by clicking the add icon. For each WAN underlay, you can optionally enable the following settings: • *Private Link*: No overlays will be created on private links.
	Network Advertisement	Configure network advertisement for the branch. Network advertisement can be set to one of the following: • *Connected*: Type the network interface to advertise. Additional interfaces can be added by clicking the add icon. • *Static*: Type the network prefix to advertise. Additional network prefixes can be added by clicking the add icon.
	Advanced	Expand to view advanced settings, including configuration of route maps for hub overlays. You can apply the route map settings to all hub overlays or specify them individually.

7. For the *Template Options*, configure the following settings and click *Next*.



Add Overlay Objects to SD-WAN Template	Optionally, you can toggle this setting ON to automatically add the overlay objects configured by this template to a new or existing SD-WAN template. Select an existing SD-WAN template or click the add icon to create a new SD-WAN template.
Add Overlay Interfaces and Zones	Optionally, you can toggle this setting ON to add overlay interfaces and zones.
Add Healthcheck Servers for Each HUB as Performance SLA	Optionally, you can toggle this setting ON to add health check servers for each hub as performance SLAs.

8. The summary window displays a summary of the SD-WAN overlay configurations that will be created by this template.
9. When you click *Finish*, multiple provisioning templates are created based on the information you provided. The templates are automatically assigned to the devices specified by the wizard.



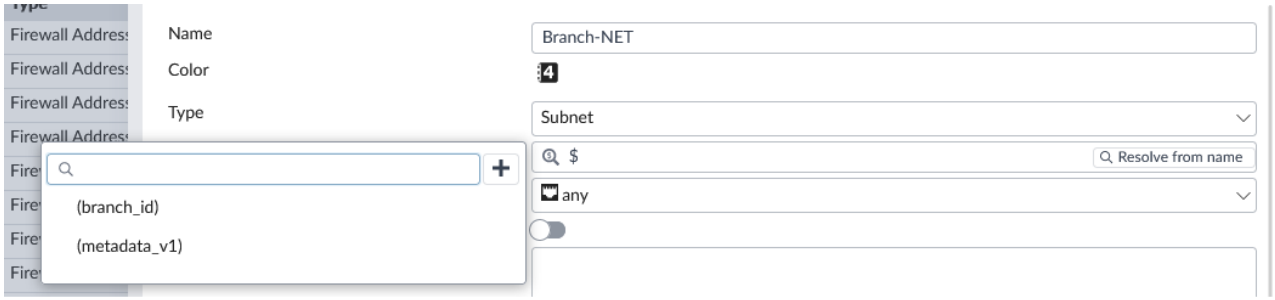
10. When complete, you can deploy the SD-WAN provisioning templates in your environment.

Metafield support on dynamic objects - FMG

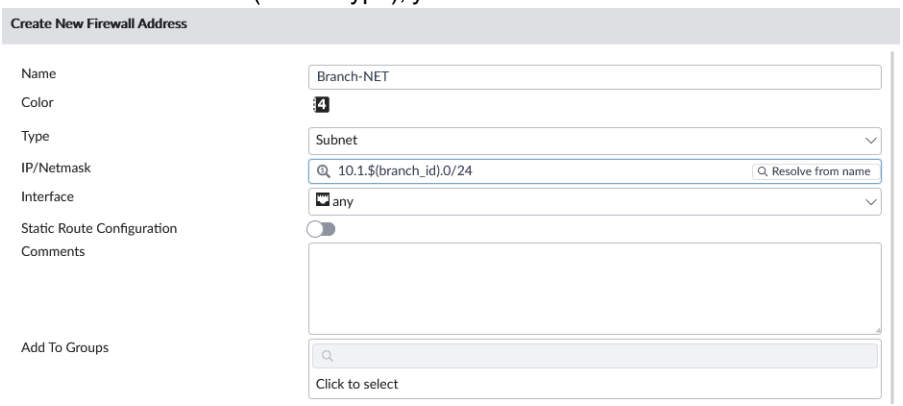
In FortiManager 7.2.0, metadata variables can be used in dynamic objects in place of per-device mappings.

To use a metadata variable in a dynamic objects:

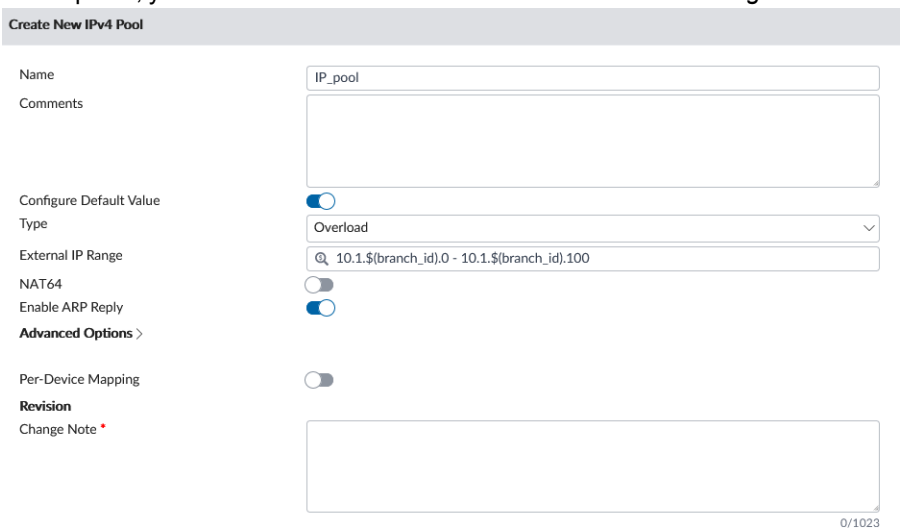
1. Go to *Policy & Objects* > *Object Configurations*.
2. Create or edit a firewall address, IP pool, or virtual IP.
3. Add the metadata in a supported text field using the following format: `$<metadata_variable_name.`
   When `$` is typed into a supported text field, available metadata variables are displayed for selection. You can click the add button to create a new metadata variable.



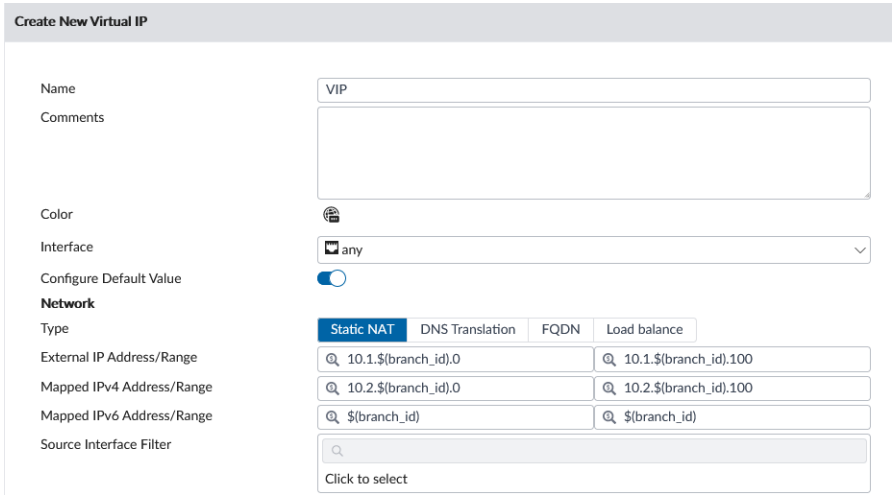
- For firewall addresses (subnet type), you can use metadata variables in the *IP/Netmask* field.



- For IP pools, you can use metadata variables in the *External IP Range* field.



- For virtual IPs, you can use metadata variables in the *External IP Address/Range*, *Mapped IPv4 Address/Range*, and *Mapped IPv6 Address/Range fields*.



Model device blueprints - FMG

In FortiManager 7.2.0, you can create device blueprints to simplify configuration of certain device settings, including device groups, configuring pre-run templates, policy packages, provisioning templates, and more. Once a device blueprint has been created, it can be selected when adding a model device or when importing multiple model devices from a CSV file.

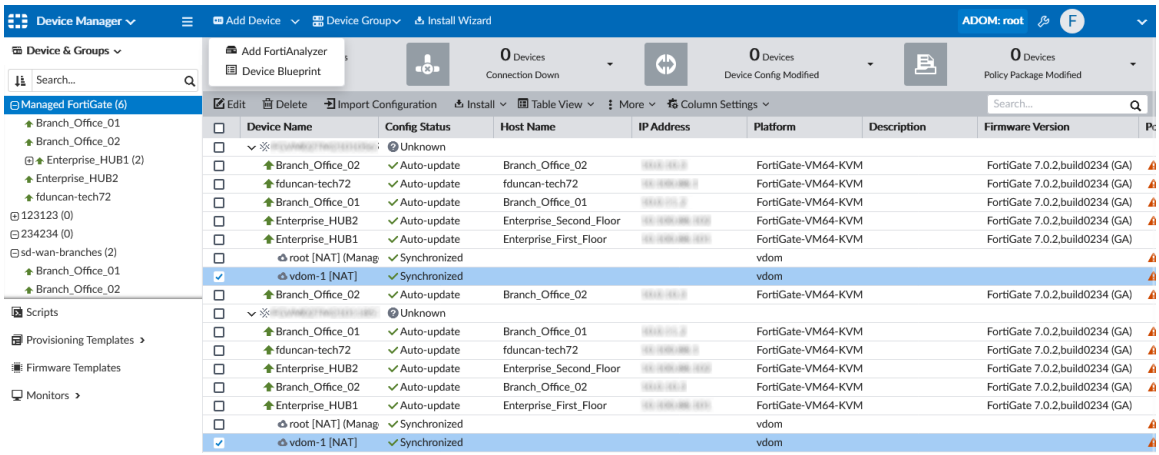
The following information is available:

- Creating a device blueprint on page 14
- Adding model devices using a blueprint on page 15

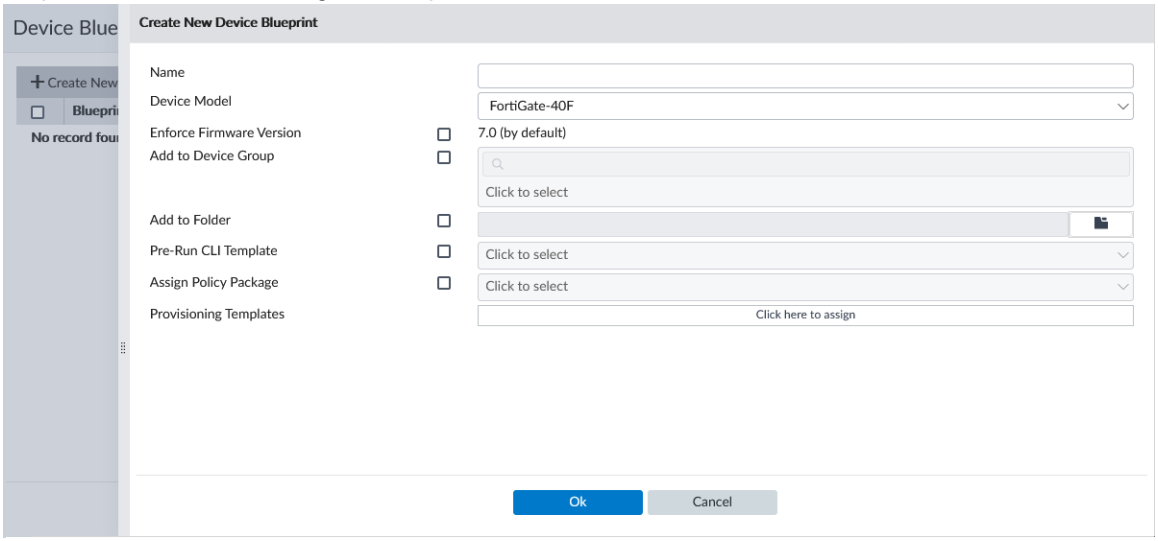
Creating a device blueprint

To create a new device blueprint:

1. Go to Device Manager, and select *Device Blueprint* from the *Add Device* dropdown menu. Previously configured blueprints are displayed in the table below and can be edited or deleted.



2. Click *Create New* to add a new blueprint.
3. Select the model devices to which the blueprint can be applied.
4. Configure the device setting details for the blueprint. For example, you can specify a device group and provisioning template for the devices using this blueprint.



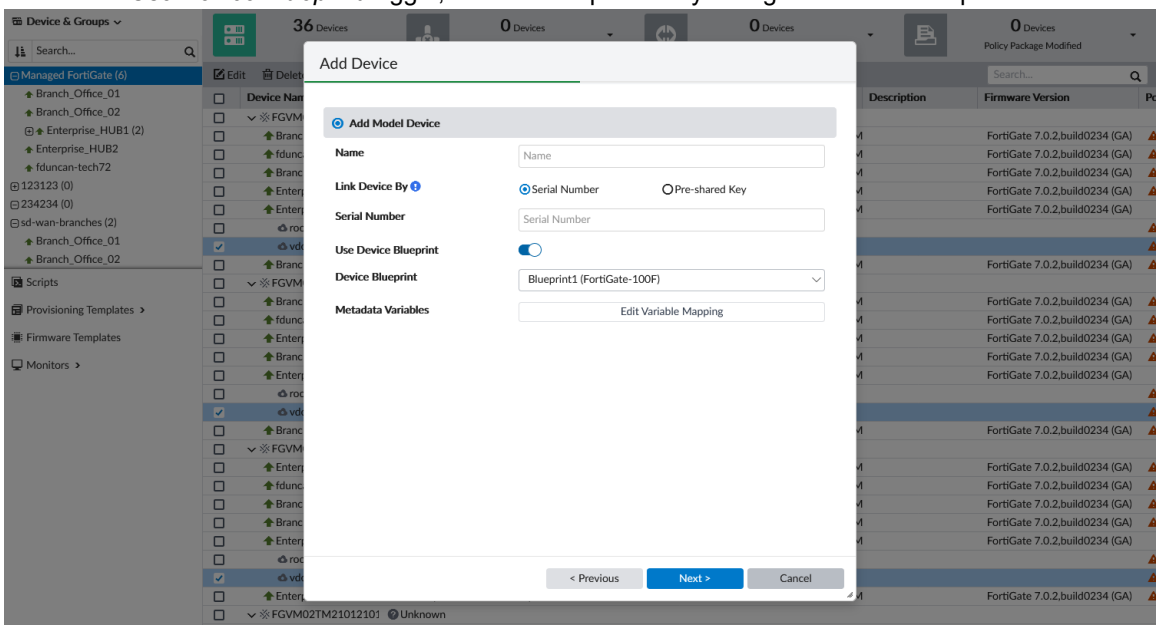
5. Click *OK*.

Adding model devices using a blueprint

To use a blueprint when adding a model device:

1. Go to *Device Manager* > *Device & Groups*.
2. Click *Add Device*. The *Add Device* wizard displays.
3. Click *Add Model Device*.
   The Add Device window is displayed.
4. Enter the name and serial number or pre-shared key for the device.

5. Enable the *Use Device Blueprint* toggle, and select a previously configured device blueprint.

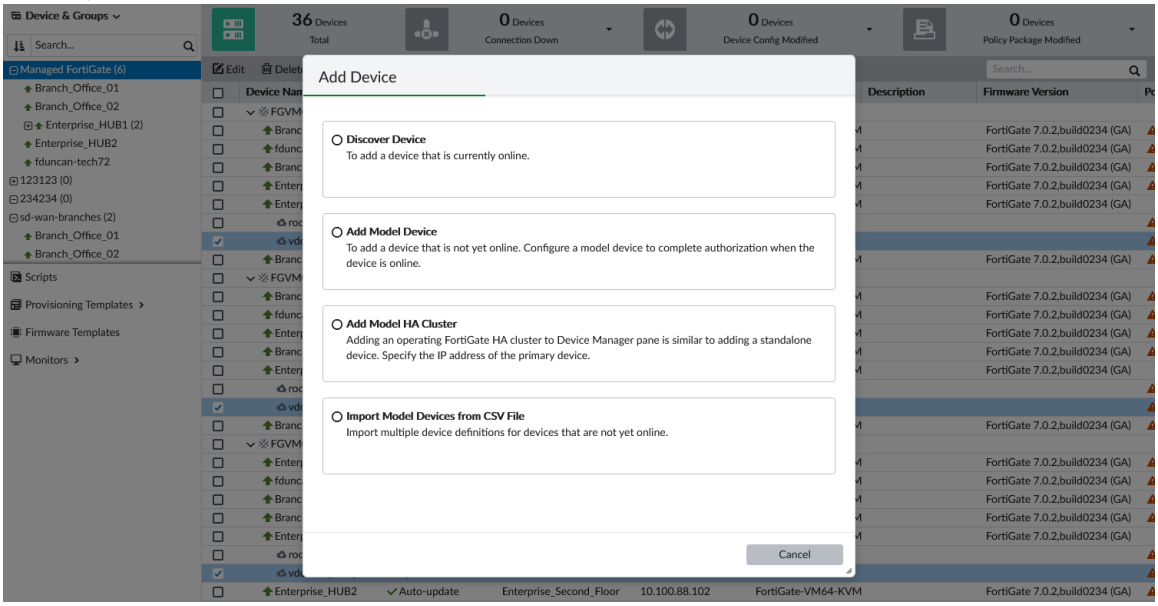


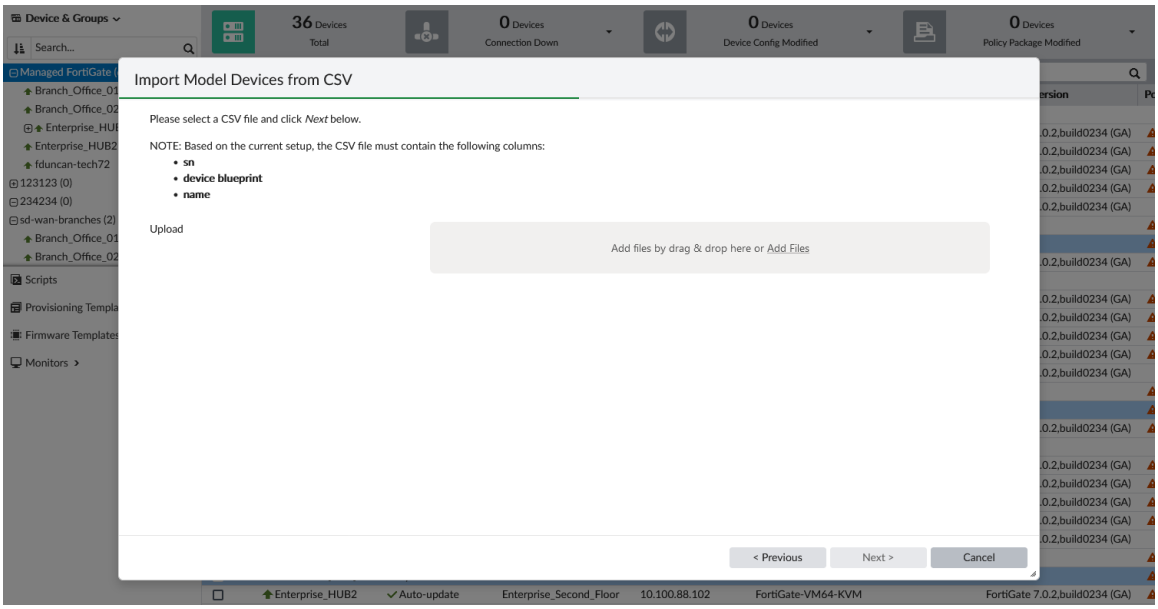
You can alternatively click the add icon to create a new device blueprint.

6. Optionally, configure the metadata variables for this device.
7. Click *Next* to continue importing the device.

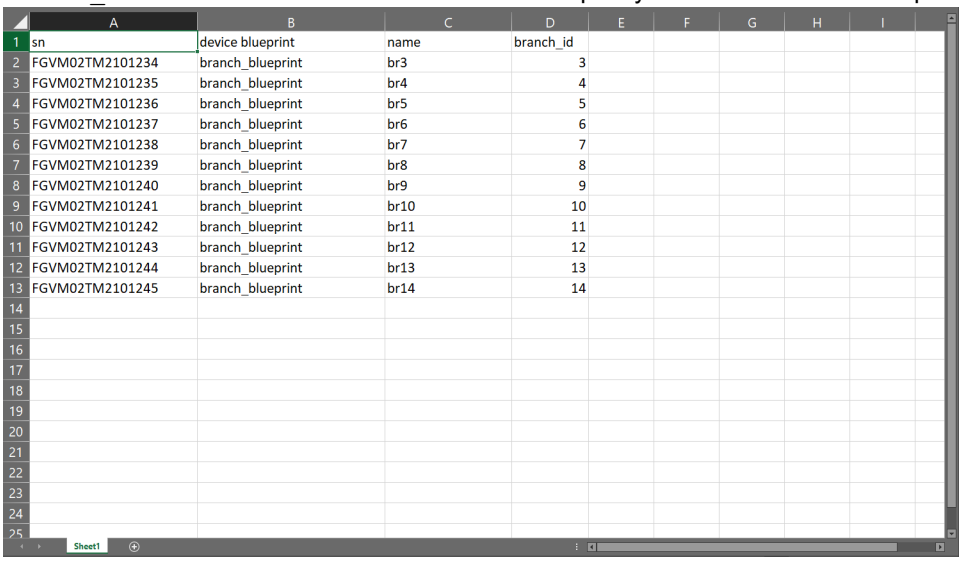
To import model devices from a CSV File:

1. If ADOMs are enabled, select the ADOM to which you want to add the device.
2. Go to *Device Manager* > *Device & Groups*.
3. Click *Add Device*.
   The Add Device window is displayed.
4. Click *Import Model Devices from CSV File*.





5. Configure your local CSV file for the devices that you want to import. CSV files must contain the following columns: `sn`, `device blueprint`, and `name`, with the respective data listed in the cells below.
   Additional columns can be added for each metadata variable that you want to specify. In the following image, the `branch_id` metadata variable has been added to specify this variable for each imported device.



	A	B	C	D
1	sn	device blueprint	name	branch_id
2	FGVM02TM2101234	branch_blueprint	br3	3
3	FGVM02TM2101235	branch_blueprint	br4	4
4	FGVM02TM2101236	branch_blueprint	br5	5
5	FGVM02TM2101237	branch_blueprint	br6	6
6	FGVM02TM2101238	branch_blueprint	br7	7
7	FGVM02TM2101239	branch_blueprint	br8	8
8	FGVM02TM2101240	branch_blueprint	br9	9
9	FGVM02TM2101241	branch_blueprint	br10	10
10	FGVM02TM2101242	branch_blueprint	br11	11
11	FGVM02TM2101243	branch_blueprint	br12	12
12	FGVM02TM2101244	branch_blueprint	br13	13
13	FGVM02TM2101245	branch_blueprint	br14	14

6. Drag and drop the CSV file into the *Upload* area, or select the CSV file location on your computer.
   The model devices' serial numbers, names, blueprints, and optional metadata variables are displayed in the table.
7. Review the device list, and click *Next* to begin importing the devices.
8. Click *Finish* when the import process is complete.

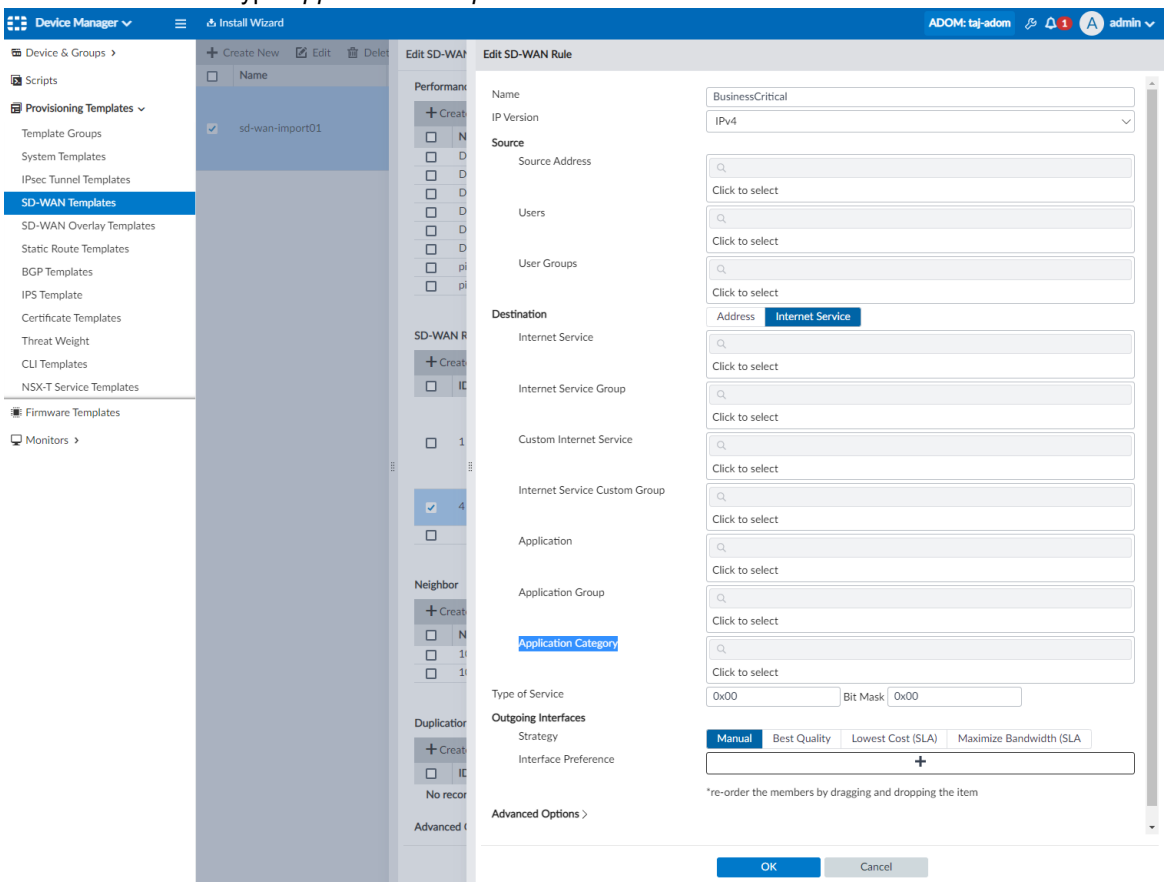
Application categories in SD-WAN rules - FMG

In FortiManager 7.2.0, the *Internet Services > Application Category* option has been added when configuring SD-WAN rules.

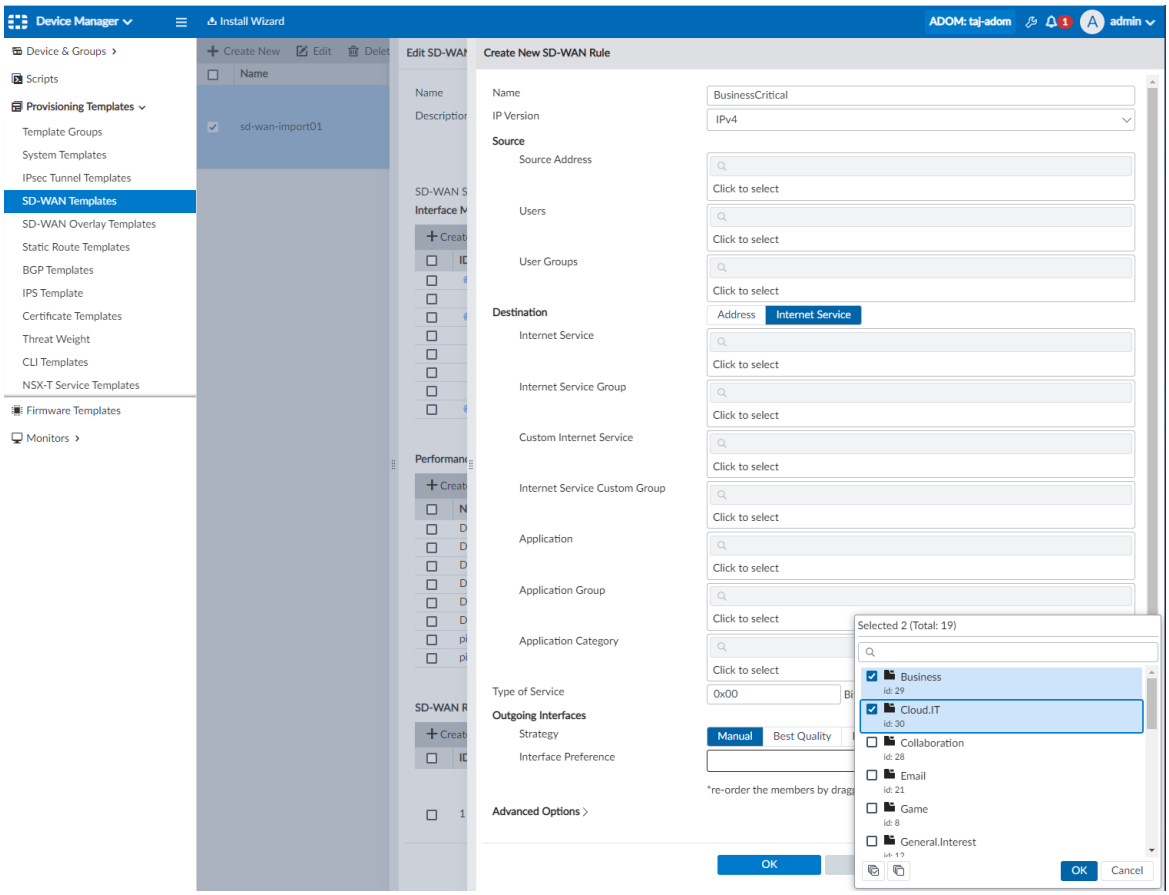
The application category uses the default internet service database (ISDB) categories received from FortiGuard. This feature is available in a FortiManager 7.2 ADOM with 7.2 or later FortiGate devices.

To configure application groups for SD-WAN rules in a template:

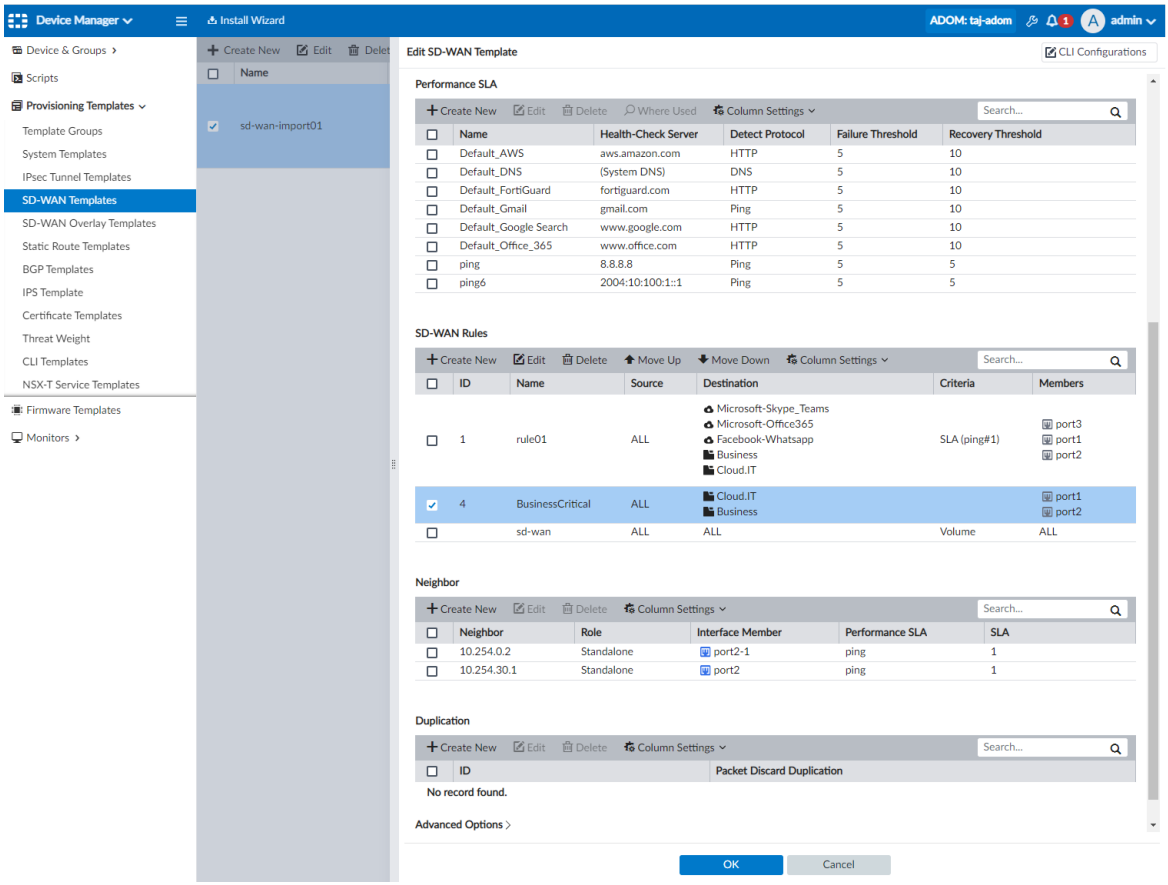
1. In FortiManager, make sure you're in a 7.2 ADOM.
2. Go to *Device Manager* > *Provisioning Templates* > *SD-WAN Templates*, and create or edit a template.
3. Under *SD-WAN Rules*, create a new rule.
4. Set the *Destination* as *Internet Service*.
   The new destination type *Application Group* has been added.



5. Select categories from the default ISDB list. New categories can be created by clicking the add button in the selection window.



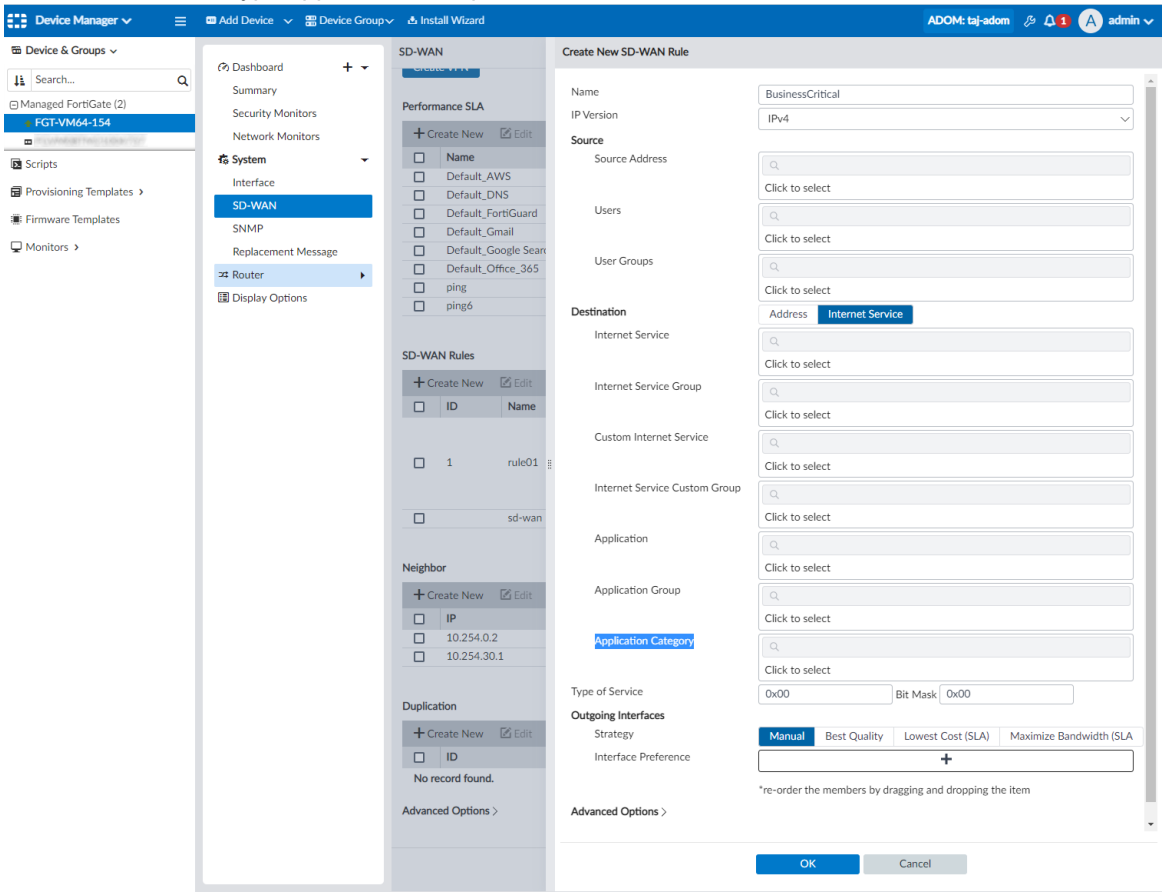
6. Click *OK* to save the SD-WAN rule.



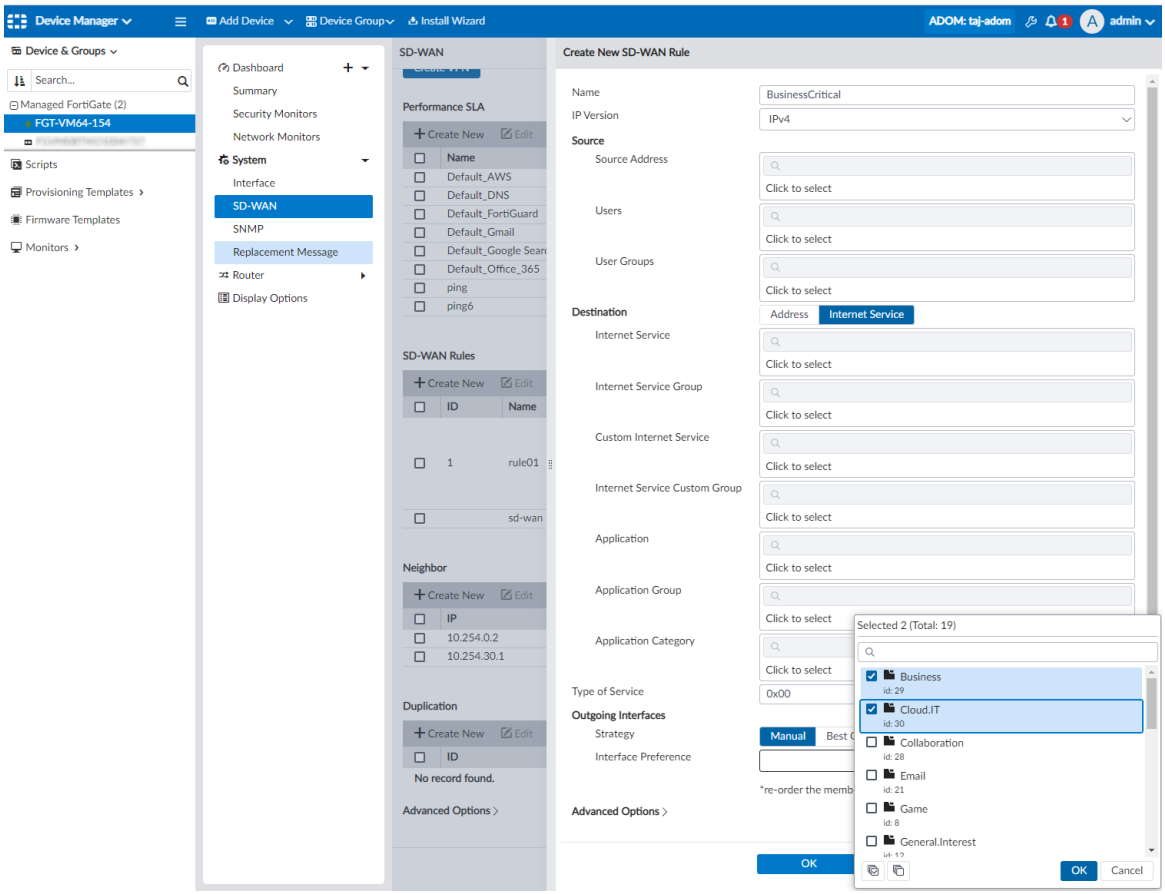
To configure application groups for SD-WAN rules in the device database:

1. In FortiManager, make sure you're in a 7.2 ADOM.
2. Go to *Device Manager* > *Device & Groups*.
3. Select a FortiGate device (7.2 or later) to manage the device database.
4. Go to *System* > *SD-WAN* > *SD-WAN Rules*, and create a new rule.

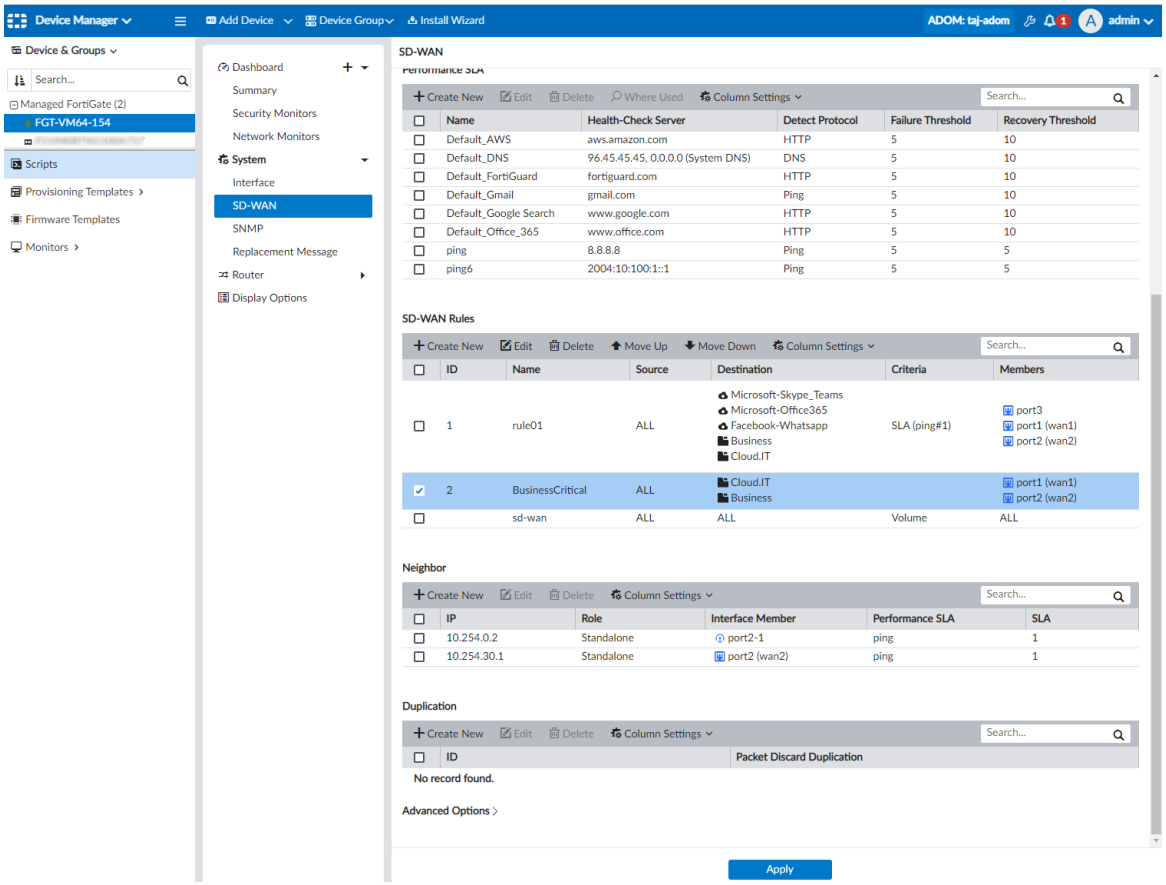
5. Set the *Destination* as Internet Service.
   The new destination type *Application Group* has been added.



6. Select categories from the default ISDB list. New categories can be created by clicking the add button in the selection window.



7. Click *OK* to save the SD-WAN rule.



Reporting

7.2.0

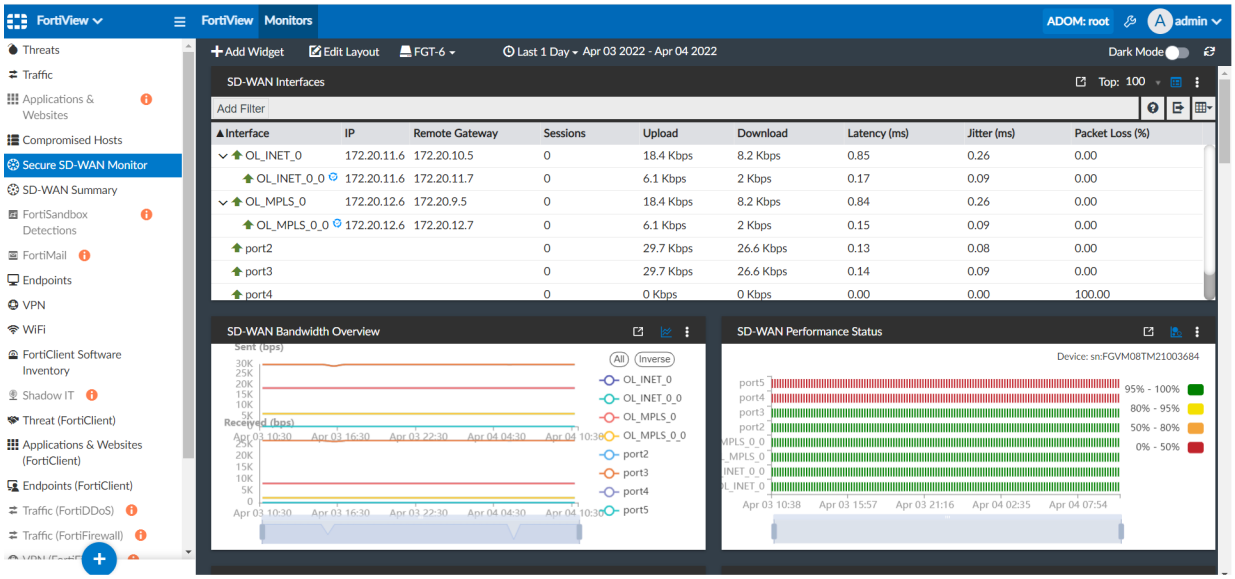
SD-WAN chart to include more ADVPN shortcut information - FAZ

The *SD-WAN Interfaces* widget is available in *FortiView* > *Monitors* > *Secure SD-WAN Monitor*.

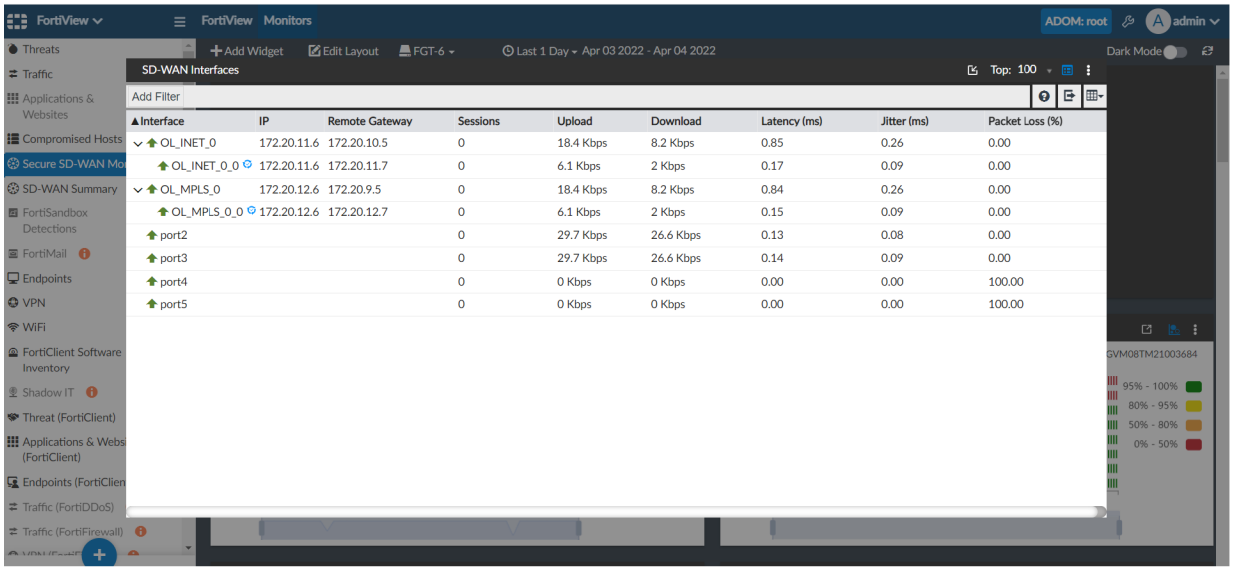
This widget displays the following information for SD-WAN interfaces: IP, Remote Gateway, Sessions, Upload, Download, Latency (ms), Jitter (ms), and Packet Loss (%). The Upload and Download columns can be used to show outbound and inbound bandwidth. For a VPN tunnel interface, IP and Remote Gateway are the local IP and Remote Gateway IP of the VPN tunnel.

To view the SD-WAN interface information:

1. Go to *FortiView* > *Monitors* > *Secure SD-WAN Monitor*.
   The SD-WAN Interfaces widget is displayed.



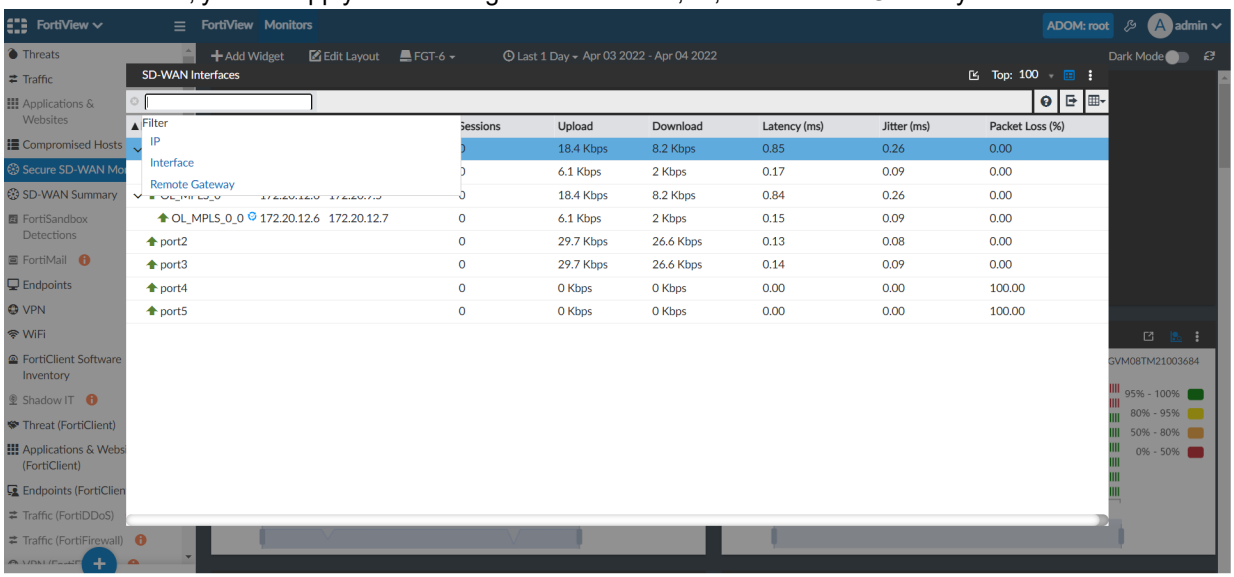
2. If there is an expand icon in the row, click the icon to view the ADVPN shortcut information in a row below. The IP and Remote Gateway are the local spoke IP and remote spoke IP of the shortcut VPN tunnel.



The following information is available in the widget:

Interface	The name of the interface.
IP	The IP address for the interface.
Remote Gateway	The remote gateway IP address.
Sessions	The number of sessions for the interface.
Upload	The upload speed for the interface.
Download	The download speed for the interface.
Latency (ms)	The latency for the interface.
Jitter (ms)	The jitter for the interface.
Packet Loss (%)	The packet loss for the interface.

3. In the table chart, you can apply the following filters: Interface, IP, and Remote Gateway.



SD-WAN chart for MOS scoring - FAZ

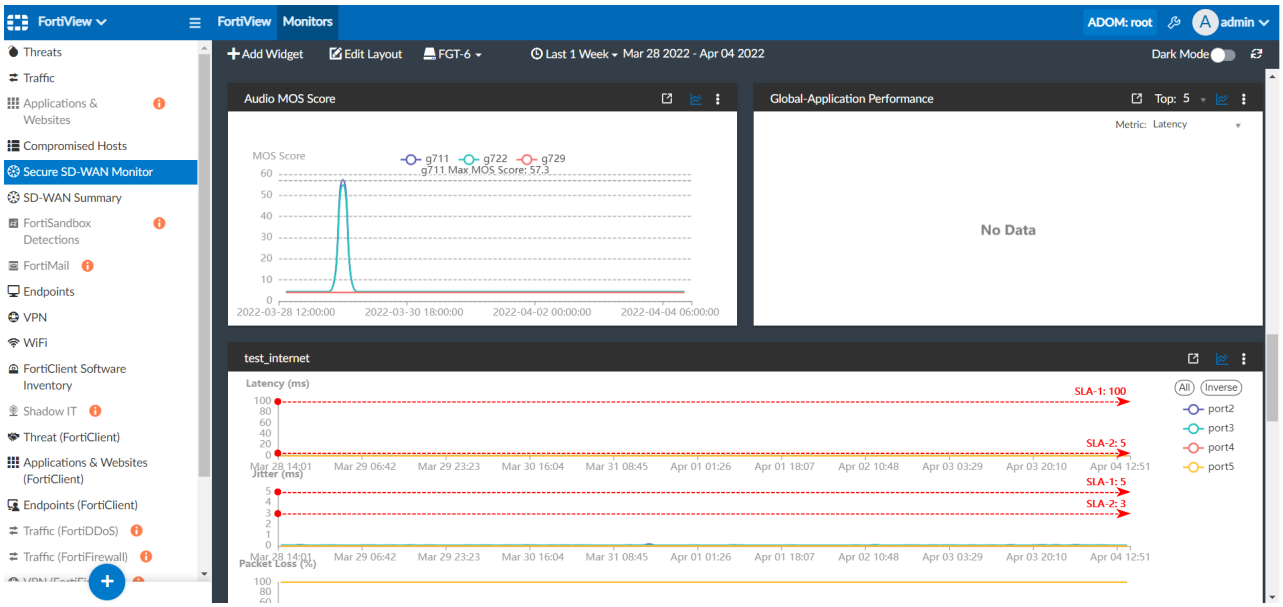
An *Audio MOS Score* widget is added to *FortiView* > *Monitors* > *Secure SD-WAN Monitor* and *FortiView* > *Monitors* > *SD-WAN Summary*. These widgets display logs for the MOS (mean opinion score) of voice and video traffic.

MOS is a method to measure the impact network quality has on the quality of a voice call. It is the industry standard for measuring voice and video quality on a WAN link.

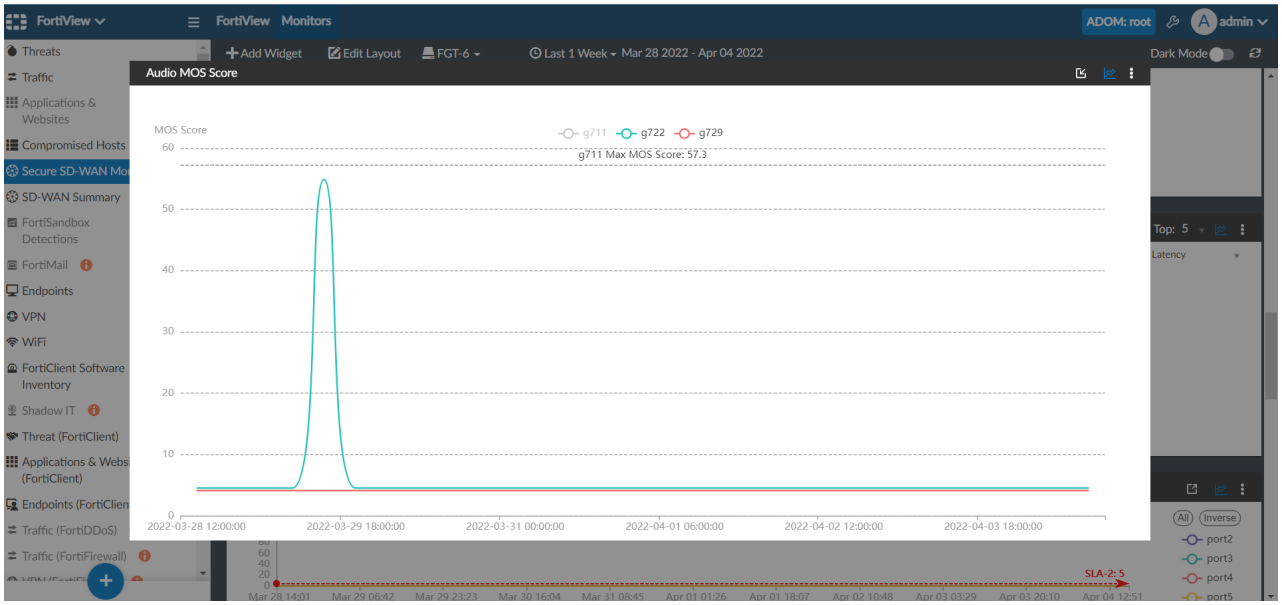
The FortiGate version must be on version 7.2 or later and have the MOS codec and MOS threshold attributes defined for SD-WAN health check in order for FortiAnalyzer to display information in the MOS scoring widgets.

To view the Audio MOS Score for individual devices:

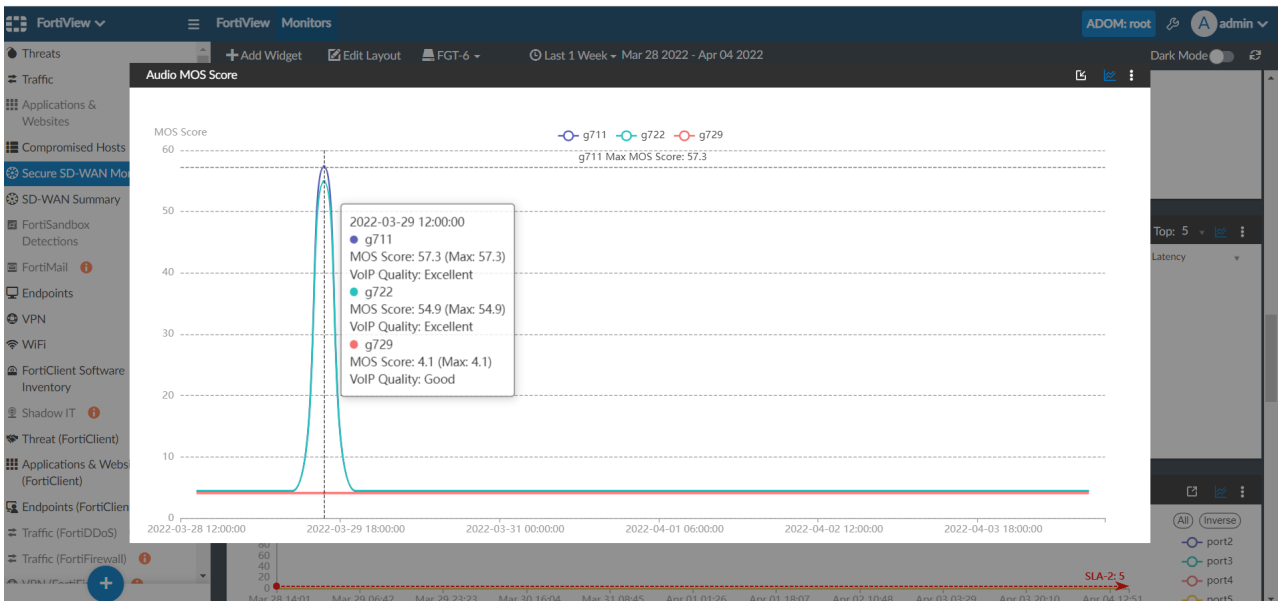
1. Go to *FortiView* > *Monitors* > *Secure SD-WAN Monitor*.
2. Click *Add Widget*, and add the *Audio MOS Score* widget.
   The widget includes a line graph of the MOS score for different codecs for the selected device over a specified time period.



3. Click a codec in the legend to make it appear/disappear on the chart.
   Greyed-out interfaces on the legend do not appear on the chart.

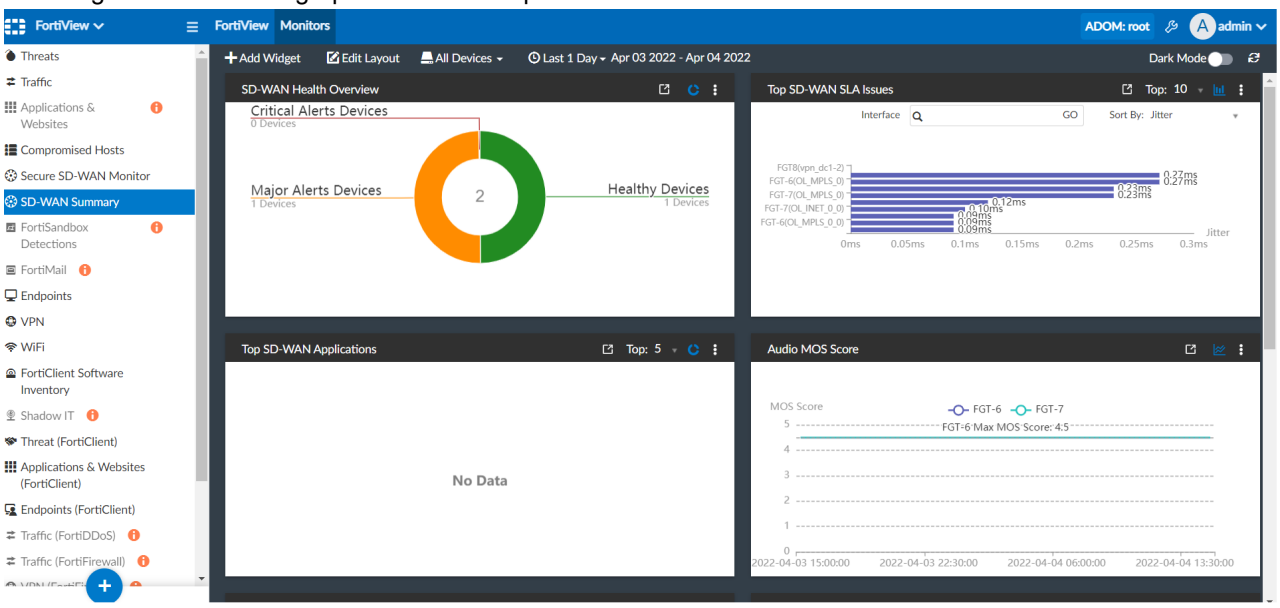


4. Hover your cursor over the chart to see a summary at that point.
   This summary includes the MOS score and the VoIP quality at that time. VoIP quality is divided into levels based on MOS scoring: Excellent = 4.3 - 5.0, Good = 4.0 - 4.3, Fair = 3.6 - 4.0, Poor = 3.1 - 3.6, Bad = 2.6 - 1.0.

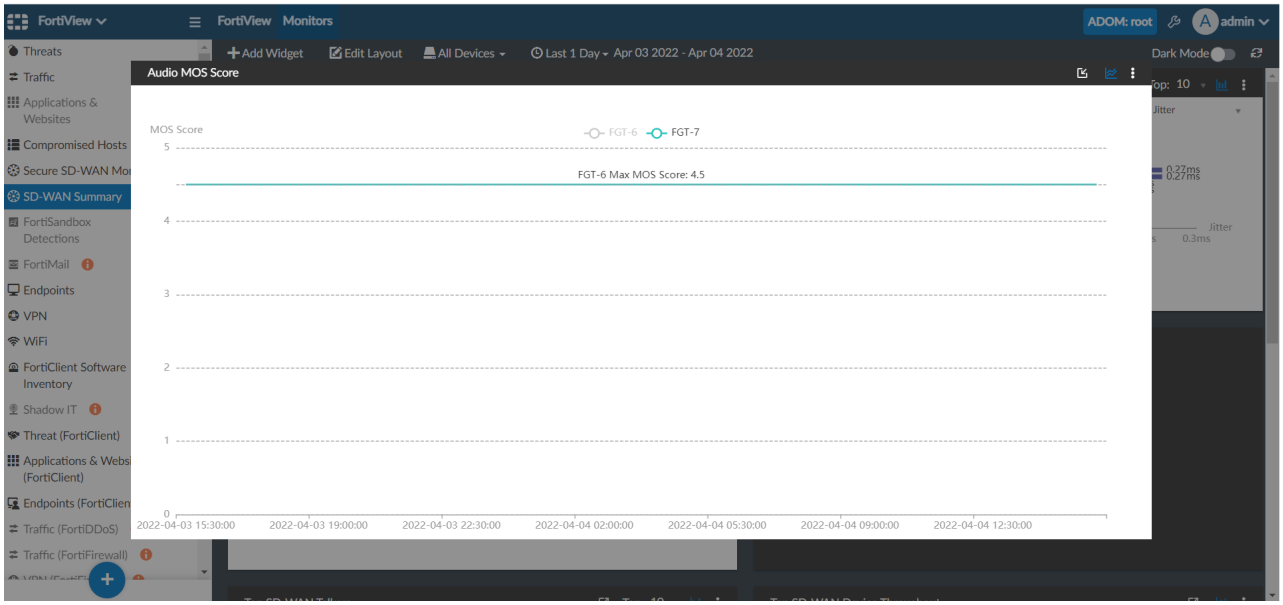


To view the Audio MOS Score across all devices:

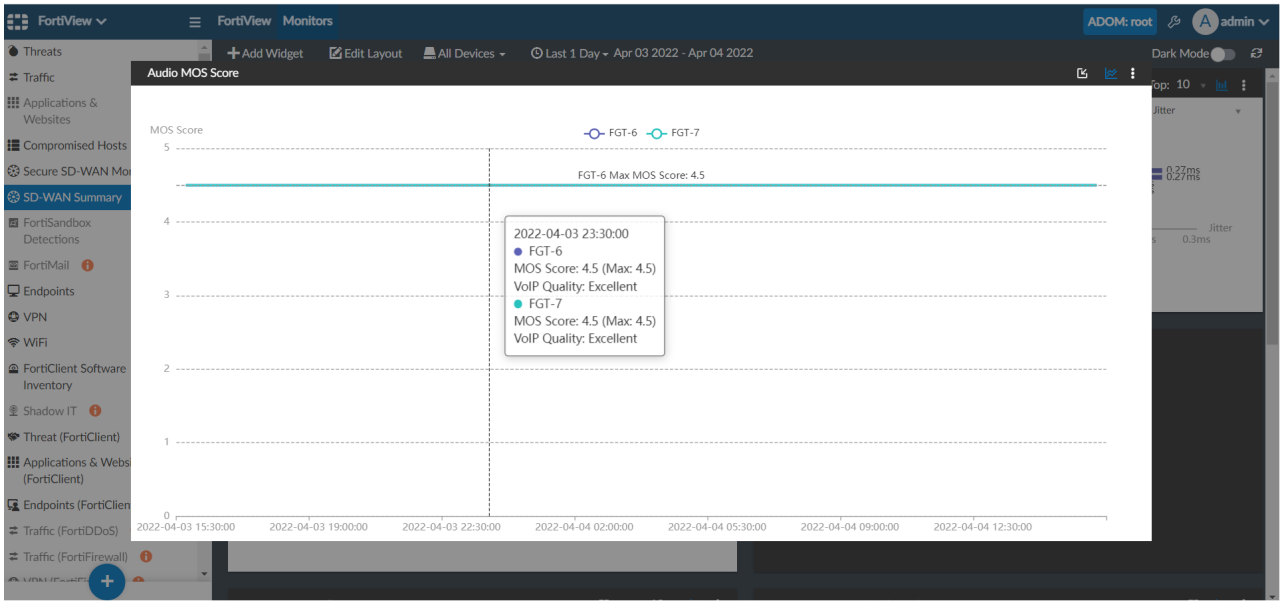
1. Go to *FortiView* > *Monitors* > *SD-WAN Summary*.
2. Click *Add Widget*, and add the *Audio MOS Score* widget.
   The widget includes a line graph of MOS score per device on the network.



3. Click a device in the legend to make it appear/disappear on the chart.
   Greyed-out devices on the legend do not appear on the chart.



4. Hover your cursor over the chart to see a summary at that point.
   This summary includes the MOS score and the VoIP quality at that time.



To configure the FortiGate MOS codec and threshold in health check settings:

1. Access the FortiGate CLI.
2. Enter the following commands:

```
config system sdwan
   config health-check
      edit <name>
         set server {string}
         set sla-fail-log-period {integer}
         set sla-pass-log-period {integer}
```

```
            set members <seq-num1>, <seq-num2>, ...
            set mos-codec [g711|g722|...]
        config sla
            edit <id>
                set link-cost-factor {option1}, {option2}, ...
                set mos-threshold {string}
            next
        end
```

For example:

```
    config system sdwan
        config health-check
            edit "test_dc"
                set server "10.200.1.1"
                set sla-fail-log-period 15
                set sla-pass-log-period 15
                set members 1 2
                set mos-codec g722
        config sla
            edit 1
                set link-cost-factor latency jitter packet-loss mos
                set mos-threshold "2.0"
            next
        end
```

Routing

7.2.0

SD-WAN segmentation over a single overlay

SD-WAN, VPN, and BGP configurations support L3 VPN segmentation over a single overlay. In these configurations, a hub and spoke SD-WAN deployment requires that branch sites, or spokes, are able to accommodate multiple companies or departments, and each company's subnet is separated by a different VRF. A subnet on one VRF cannot communicate with a subnet on another VRF between different branches, but can communicate with the same VRF.

New SD-WAN options

VRF-aware SD-WAN health checks

SD-WAN on the originating spoke can tag the health check probes with the correct VRF when transmitting to a multi-VRF tunnel. The hub can then forward the probes to the correct health check server in the same VRF as the hub.

```
config system sdwan
   config health-check
       edit <name>
           set vrf <vrf id>
           set source <address>
       next
   end
end
```

`vrf <vrf id>`	Virtual Routing Forwarding ID.
`source <address>`	Source IP address used in the health-check packet to the server.

Overlay stickiness

When a hub has multiple overlays, traffic received on one overlay should egress on the same overlay when possible. The `service-sla-tie-break` option ensures overlay stickiness. In SD-WAN service rules, options are available to ensure that traffic received in a zone stays in that zone.

```
config system sdwan
    config zone
        edit <name>
            set service-sla-tie-break input-device
        next
    end
    config service
        edit <id>
            set input-zone <zone>
            set tie-break input-device
        next
    end
end
```

`service-sla-tie-break input-device`	Members that meet the SLA are selected by matching the input device.
`input-zone <zone>`	Source input-zone name.
`tie-break input-device`	Members that meet the SLA are selected by matching the input device.

New IPsec options

Configurable rate limit for shortcut offers sent by the hub

By default, the hub sends a shortcut offer to a spoke every five seconds. If the hub continues to send offers that keep failing, and there are a large number of spokes, this can cause a high load on the hub. This setting makes the interval between shortcut offers configurable.

```
config vpn ipsec phase1-interface
    edit <name>
        set auto-discovery-offer-interval <interval>
    next
end
```

`auto-discovery-offer-interval <interval>`	Interval between shortcut offer messages, in seconds (1 - 300, default = 5).

Segmentation over a single overlay

Segmentation requires that VRF info is encapsulated within the IPsec VPN tunnel. This setting enables multi-VRF IPSEC tunnels.

```
config vpn ipsec phase1-interface
    edit <name>
        set encapsulation vpn-id-ipip
    next
end
```

`encapsulation vpn-id-ipip`	VPN ID with IPIP encapsulation.

New VPN configuration for BGP

The role of a VRF can be specified, along with other VRF details. Up to 64 VRFs can be configured per VDOM for devices that support 200 VDOMs.

```
config router bgp
    config vrf
        edit <vrf>
            set role {standalone | ce | pe}
            set rd <string>
            set export-rt <route_target>
            set import-rt <route_target>
            set import-route-map <route_map>
            config leak-target
                edit <vrf>
                    set route-map <route-map>
                    set interface <interface>
                next
            end
        next
    end
end
```

`role {standalone | ce | pe}`	VRF role: standalone, customer edge (CE), or provider edge (PE).
`rd <string>`	Route Distinguisher: AA|AA:NN. This option is only available when the role is CE.
`export-rt <route_target>`	List of export route target. This option is only available when the role is CE.
`import-rt <route_target>`	List of import route target. This option is only available when the role is CE.
`import-route-map <route_map>`	Import route map. This option is only available when the role is CE.
`route-map <route-map>`	Route map of VRF leaking.
`interface <interface>`	Interface that is used to leak routes to the target VRF.

In FortiOS 7.0, `config vrf` was `config vrf-leak`, and `config leak-target` was `config target`.

Display BGP routes by VRF and neighbor

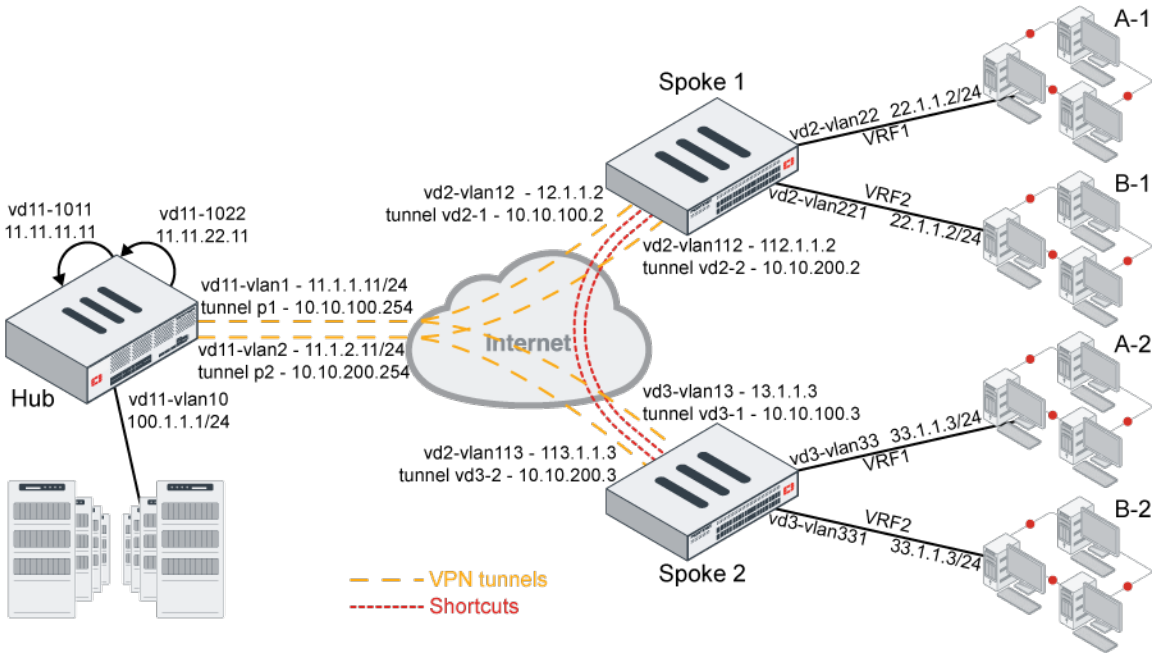
```
# diagnose ip router bgp set-filter vrf <vrf>
# diagnose ip router bgp set-filter neighbor <neighbor address>
# diagnose ip router bgp set-filter reset
# execute router clear bgp vpnv4 unicast soft {in | out}
# get router info filter show
# get router info filter vrf {vrf | all}
```

Examples

In example 1, multiple companies (or departments of a company) share the ADVPN. Company A and company B each have two branches in two different locations. Company A's branches (A-1 and A-2) can talk to each other using the VPN shortcut, but not to company B's branches (B-1 and B-2). Likewise, company B's branches can talk to each other using the VPN shortcut, but not to company A's branches. Traffic can share the tunnels and shortcuts, but cannot be mixed up.

Example 2 shows that performance SLA health checks can be sent from a spoke's VRF to the loopback on the hub that is in the same VRF.

Example 3 shows that when traffic is ingress on the hub on one overlay, it will preferably egress on the same overlay.



Example 1

In this example, two spokes each have two tunnels to the hub.

- Each spoke has two VRFs behind it that can use the same IP address or subnets.
- The computers in VRF1 behind spoke 1 can talk to the computers in VRF1 behind spoke 2, but not to any of the computers in the VRF2s behind either spoke.
- The computers in VRF2 behind spoke 1 can talk to the computers in VRF2 behind spoke 2, but not to any of the computers in the VRF1s behind either spoke.

To configure the hub:

```
config router bgp
    set as 65505
    set router-id 11.11.11.11
    set ibgp-multipath enable
    set additional-path enable
    set additional-path-vpnv4 enable
    set cluster-id 11.12.13.14
```

```
set additional-path-select 3
config neighbor-group
    edit "gr1"
        set capability-graceful-restart enable
        set capability-default-originate enable
        set next-hop-self-rr enable
        set soft-reconfiguration-vpnv4 enable
        set remote-as 65505
        set additional-path both
        set additional-path-vpnv4 both
        set adv-additional-path 3
        set route-reflector-client enable
        set route-reflector-client-vpnv4 enable
    next
    edit "gr2"
        set capability-graceful-restart enable
        set capability-default-originate enable
        set next-hop-self-rr enable
        set soft-reconfiguration-vpnv4 enable
        set remote-as 65505
        set additional-path both
        set additional-path-vpnv4 both
        set adv-additional-path 3
        set route-reflector-client enable
        set route-reflector-client-vpnv4 enable
    next
end
config neighbor-range
    edit 1
        set prefix 10.10.100.0 255.255.255.0
        set neighbor-group "gr1"
    next
    edit 2
        set prefix 10.10.200.0 255.255.255.0
        set neighbor-group "gr2"
    next
end
config network
    edit 12
        set prefix 11.11.11.11 255.255.255.255
    next
    edit 22
        set prefix 11.11.22.11 255.255.255.255
    next
    edit 10
        set prefix 100.1.1.0 255.255.255.0
    next
    edit 33
        set prefix 11.1.1.0 255.255.255.0
    next
end
config vrf
    edit "0"
        set role pe
    next
    edit "1"
```

```
            set role ce
            set rd "1:1"
            set export-rt "1:1"
            set import-rt "1:1"
        next
        edit "2"
            set role ce
            set rd "2:1"
            set export-rt "2:1"
            set import-rt "2:1"
        next
    end
end
config vpn ipsec phase1-interface
    edit "p1"
        set type dynamic
        set interface "vd11-vlan1"
        set peertype any
        set net-device disable
        set proposal aes128-sha1
        set add-route disable
        set dpd on-idle
        set dhgrp 5
        set auto-discovery-sender enable
        set auto-discovery-offer-interval 10
        set encapsulation vpn-id-ipip
        set psksecret **********
        set dpd-retryinterval 60
    next
    edit "p2"
        set type dynamic
        set interface "vd11-vlan2"
        set peertype any
        set net-device disable
        set proposal aes128-sha1
        set add-route disable
        set dpd on-idle
        set dhgrp 5
        set auto-discovery-sender enable
        set auto-discovery-offer-interval 10
        set encapsulation vpn-id-ipip
        set psksecret **********
        set dpd-retryinterval 60
    next
end
config vpn ipsec phase2-interface
    edit "p1"
        set phase1name "p1"
        set proposal aes128-sha1
        set dhgrp 5
    next
    edit "p2"
        set phase1name "p2"
        set proposal aes128-sha1
        set dhgrp 5
```

```
    next
end
```

To configure a spoke:

```
config router bgp
    set as 65505
    set router-id 2.2.2.2
    set ebgp-multipath enable
    set ibgp-multipath enable
    set network-import-check disable
    set additional-path enable
    set additional-path6 enable
    set additional-path-vpnv4 enable
    set recursive-next-hop enable
    set graceful-restart enable
    set additional-path-select 4
    config neighbor
        edit "10.10.100.254"
            set capability-dynamic enable
            set capability-graceful-restart-vpnv4 enable
            set soft-reconfiguration enable
            set soft-reconfiguration-vpnv4 enable
            set remote-as 65505
            set additional-path both
            set additional-path-vpnv4 both
            set adv-additional-path 3
        next
        edit "10.10.200.254"
            set capability-dynamic enable
            set capability-graceful-restart-vpnv4 enable
            set soft-reconfiguration enable
            set soft-reconfiguration-vpnv4 enable
            set remote-as 65505
            set additional-path both
            set additional-path-vpnv4 both
            set adv-additional-path 3
        next
    end
    config network
        edit 3
            set prefix 22.1.1.0 255.255.255.0
        next
        edit 4
            set prefix 12.12.12.0 255.255.255.0
        next
    end
    config vrf
        edit "0"
            set role pe
        next
        edit "1"
            set role ce
            set rd "1:1"
            set export-rt "1:1"
            set import-rt "1:1"
```

```
        next
        edit "2"
            set role ce
            set rd "2:1"
            set export-rt "2:1"
            set import-rt "2:1"
        next
    end
end
config vpn ipsec phase1-interface
    edit "vd2-1"
        set interface "vd2-vlan12"
        set peertype any
        set net-device enable
        set proposal aes128-sha1
        set add-route disable
        set dhgrp 5
        set idle-timeout enable
        set idle-timeoutinterval 5
        set auto-discovery-receiver enable
        set encapsulation vpn-id-ipip
        set remote-gw 11.1.1.11
        set psksecret **********
    next
    edit "vd2-2"
        set interface "vd2-vlan112"
        set peertype any
        set net-device enable
        set proposal aes128-sha1
        set add-route disable
        set dhgrp 5
        set auto-discovery-receiver enable
        set encapsulation vpn-id-ipip
        set remote-gw 11.1.2.11
        set psksecret **********
    next
end
config vpn ipsec phase2-interface
    edit "vd2-1"
        set phase1name "vd2-1"
        set proposal aes128-sha1
        set dhgrp 5
        set auto-negotiate enable
    next
    edit "vd2-2"
        set phase1name "vd2-2"
        set proposal aes128-sha1
        set dhgrp 5
        set auto-negotiate enable
    next
end
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
```

```
        next
        edit "SASE"
        next
        edit "zon2"
        next
    end
    config members
        edit 1
            set interface "vd2-1"
            set cost 10
        next
        edit 2
            set interface "vd2-2"
            set cost 20
        next
    end
    config health-check
        edit "ping"
            set server "11.11.11.11"
            set members 1 2
            config sla
                edit 1
                    set latency-threshold 200
                    set jitter-threshold 50
                next
            end
        next
        edit "1"
            set server "22.1.1.2"
            set vrf 1
            set members 1 2
        next
    end
    config service
        edit 2
            set mode sla
            set dst "100-200"
            config sla
                edit "ping"
                    set id 1
                next
            end
            set priority-members 2
            set use-shortcut-sla disable
        next
        edit 1
            set name "test-tag"
            set mode sla
            set dst "001-100"
            config sla
                edit "ping"
                    set id 1
                next
            end
            set priority-members 1 2
        next
```

```
    end
end
```

To check the spoke 1 routes:

```
# get router info routing-table bgp
Routing table for VRF=0
B*      0.0.0.0/0 [200/0] via 10.10.100.254 (recursive via vd2-1 tunnel 11.1.1.11 vrf 0),
04:42:57, [1/0]
                  [200/0] via 10.10.200.254 (recursive via vd2-2 tunnel 11.1.2.11 vrf 0),
04:42:57, [1/0]
B       1.1.1.1/32 [200/0] via 11.1.1.1 [2] (recursive via 12.1.1.1, vd2-vlan12), 04:42:57,
[1/0]
B       1.222.222.222/32 [200/0] via 11.1.1.1 [2] (recursive via 12.1.1.1, vd2-vlan12),
04:42:57, [1/0]
B       11.11.11.11/32 [200/0] via 10.10.100.254 (recursive via vd2-1 tunnel 11.1.1.11 vrf
0), 04:42:57, [1/0]
                       [200/0] via 10.10.200.254 (recursive via vd2-2 tunnel 11.1.2.11 vrf
0), 04:42:57, [1/0]
B       33.1.1.0/24 [200/0] via 10.10.100.254 [2] (recursive via vd2-1 tunnel 11.1.1.11 vrf
0), 04:42:57, [1/0]
                    [200/0] via 10.10.200.254 [2] (recursive via vd2-2 tunnel 11.1.2.11 vrf
0), 04:42:57, [1/0]
B       100.1.1.0/24 [200/0] via 10.10.100.254 (recursive via vd2-1 tunnel 11.1.1.11 vrf 0),
04:42:57, [1/0]
                     [200/0] via 10.10.200.254 (recursive via vd2-2 tunnel 11.1.2.11 vrf 0),
04:42:57, [1/0]

Routing table for VRF=1
B V     33.1.1.0/24 [200/0] via 10.10.100.3 [2] (recursive via vd2-1 tunnel 11.1.1.11 vrf
0), 04:42:57, [1/0]
                    [200/0] via 10.10.200.3 [2] (recursive is directly connected, vd2-2_0),
04:42:57, [1/0]

Routing table for VRF=2
B V     33.1.1.0/24 [200/0] via 10.10.100.3 [2] (recursive via vd2-1 tunnel 11.1.1.11 vrf
0), 04:42:56, [1/0]
                    [200/0] via 10.10.200.3 [2] (recursive is directly connected, vd2-2_0),
04:42:56, [1/0]
```

VRF1 routes:

```
# get router info filter vrf 1
# get router info routing-table bgp
Routing table for VRF=1
B V     33.1.1.0/24 [200/0] via 10.10.100.3 [2] (recursive via vd2-1 tunnel 11.1.1.11 vrf
0), 04:44:11, [1/0]
                    [200/0] via 10.10.200.3 [2] (recursive is directly connected, vd2-2_0),
04:44:11, [1/0]
```

To test the configuration on shortcut 1:

1. From VRF1 of spoke 1 ping VRF1 of spoke 2 and from VRF2 of spoke 1 ping VRF2 spoke 2. Both VRF1 and VRF2 source and destination IP addresses are the same, so you can see how the traffic is isolated
2. Check sessions on spoke 1:

   The output `vd=<vdom ID>:<VRF ID>` indicates that sessions are created in and stay in the corresponding VRFs.

- User at 22.1.1.22 in VRF1 on spoke 1 pings 33.1.1.33 in VRF1 on spoke2.

```
# diagnose sys session list
session info: proto=1 proto_state=00 duration=21 expire=42 timeout=0 flags=00000000
socktype=0 sockport=0 av_idx=0 use=3
origin-shaper=
reply-shaper=
per_ip_shaper=
class_id=0 ha_id=0 policy_dir=0 tunnel=/ vlan_cos=0/255
state=may_dirty
statistic(bytes/packets/allow_err): org=420/5/1 reply=420/5/1 tuples=2
tx speed(Bps/kbps): 0/0 rx speed(Bps/kbps): 0/0
orgin->sink: org pre->post, reply pre->post dev=89->131/131->89
gwy=10.10.200.3/22.1.1.22
hook=pre dir=org act=noop 22.1.1.22:48417->33.1.1.33:8(0.0.0.0:0)
hook=post dir=reply act=noop 33.1.1.33:48417->22.1.1.22:0(0.0.0.0:0)
src_mac=02:4c:a5:fc:6a:7f
misc=0 policy_id=1 pol_uuid_idx=566 auth_info=0 chk_client_info=0 vd=1:1
serial=00092eee tos=ff/ff app_list=0 app=0 url_cat=0
sdwan_mbr_seq=0 sdwan_service_id=1
rpdb_link_id=ff000001 ngfwid=n/a
npu_state=0x5040001 no_offload
no_ofld_reason:  disabled-by-policy non-npu-intf
```

- User at 22.1.1.22 in VRF2 on spoke 1 pings 33.1.1.33 in VRF2 on spoke2:

```
# diagnose sys session list
session info: proto=1 proto_state=00 duration=4 expire=56 timeout=0 flags=00000000
socktype=0 sockport=0 av_idx=0 use=3
origin-shaper=
reply-shaper=
per_ip_shaper=
class_id=0 ha_id=0 policy_dir=0 tunnel=/ vlan_cos=0/255
state=may_dirty
statistic(bytes/packets/allow_err): org=168/2/1 reply=168/2/1 tuples=2
tx speed(Bps/kbps): 39/0 rx speed(Bps/kbps): 39/0
orgin->sink: org pre->post, reply pre->post dev=113->131/131->113
gwy=10.10.200.3/22.1.1.22
hook=pre dir=org act=noop 22.1.1.22:55841->33.1.1.33:8(0.0.0.0:0)
hook=post dir=reply act=noop 33.1.1.33:55841->22.1.1.22:0(0.0.0.0:0)
src_mac=02:4c:a5:fc:6a:7f
misc=0 policy_id=1 pol_uuid_idx=566 auth_info=0 chk_client_info=0 vd=1:2
serial=00092f77 tos=ff/ff app_list=0 app=0 url_cat=0
sdwan_mbr_seq=0 sdwan_service_id=1
rpdb_link_id=ff000001 ngfwid=n/a
npu_state=0x5040001 no_offload
no_ofld_reason:  disabled-by-policy non-npu-intf
```

3. Check sessions on spoke 2:

   The output `vd=<vdom ID>:<VRF ID>` indicates that sessions are created in and stay in the corresponding VRFs.

   - User at 22.1.1.22 in VRF1 on spoke 1 pings 33.1.1.33 in VRF1 on spoke 2:

```
# diagnose sys session list
session info: proto=1 proto_state=00 duration=11 expire=49 timeout=0 flags=00000000
socktype=0 sockport=0 av_idx=0 use=3
origin-shaper=
reply-shaper=
```

```
per_ip_shaper=
class_id=0 ha_id=0 policy_dir=0 tunnel=/ vlan_cos=0/255
state=may_dirty npu
statistic(bytes/packets/allow_err): org=168/2/1 reply=168/2/1 tuples=2
tx speed(Bps/kbps): 14/0 rx speed(Bps/kbps): 14/0
orgin->sink: org pre->post, reply pre->post dev=132->92/92->132
gwy=33.1.1.33/10.10.200.2
hook=pre dir=org act=noop 22.1.1.22:27733->33.1.1.33:8(0.0.0.0:0)
hook=post dir=reply act=noop 33.1.1.33:27733->22.1.1.22:0(0.0.0.0:0)
misc=0 policy_id=1 pol_uuid_idx=630 auth_info=0 chk_client_info=0 vd=6:1
serial=000a29fd tos=ff/ff app_list=0 app=0 url_cat=0
rpdb_link_id=00000000 ngfwid=n/a
npu_state=0x4000001 no_offload
npu info: flag=0x00/0x82, offload=0/0, ips_offload=0/0, epid=0/0, ipid=0/0,
vlan=0x0000/0x0000
vlifid=0/0, vtag_in=0x0000/0x0000 in_npu=0/0, out_npu=0/0, fwd_en=0/0, qid=0/0
no_ofld_reason:  disabled-by-policy
```

- User at 22.1.1.22 in VRF2 on spoke 1 pings 33.1.1.33 in VRF2 on spoke 2:

```
# diagnose sys session list
session info: proto=1 proto_state=00 duration=17 expire=43 timeout=0 flags=00000000
socktype=0 sockport=0 av_idx=0 use=3
origin-shaper=
reply-shaper=
per_ip_shaper=
class_id=0 ha_id=0 policy_dir=0 tunnel=/ vlan_cos=0/255
state=may_dirty npu
statistic(bytes/packets/allow_err): org=168/2/1 reply=168/2/1 tuples=2
tx speed(Bps/kbps): 9/0 rx speed(Bps/kbps): 9/0
orgin->sink: org pre->post, reply pre->post dev=132->115/115->132
gwy=33.1.1.33/10.10.200.2
hook=pre dir=org act=noop 22.1.1.22:24917->33.1.1.33:8(0.0.0.0:0)
hook=post dir=reply act=noop 33.1.1.33:24917->22.1.1.22:0(0.0.0.0:0)
dst_mac=02:4c:a5:fc:6a:7f
misc=0 policy_id=1 pol_uuid_idx=630 auth_info=0 chk_client_info=0 vd=6:2
serial=000a29ca tos=ff/ff app_list=0 app=0 url_cat=0
rpdb_link_id=00000000 ngfwid=n/a
npu_state=0x4000001 no_offload
npu info: flag=0x00/0x82, offload=0/0, ips_offload=0/0, epid=0/0, ipid=0/0,
vlan=0x0000/0x0000
vlifid=0/0, vtag_in=0x0000/0x0000 in_npu=0/0, out_npu=0/0, fwd_en=0/0, qid=0/0
no_ofld_reason:  disabled-by-policy
```

To test the configuration on shortcut 2:

1. From VRF1 of spoke 1 ping VRF1 of spoke 2 and from VRF2 of spoke 1 ping VRF2 spoke 2. Both VRF1 and VRF2 source and destination IP addresses are the same, so you can see how the traffic is isolated
2. Check sessions on spoke 1:

   The output `vd=<vdom ID>:<VRF ID>` indicates that sessions are created in and stay in the corresponding VRFs.

   - User at 22.1.1.22 in VRF1 on spoke 1 pings 33.1.1.133 in VRF1 on spoke 2:

```
# diagnose sys session listsession info: proto=1 proto_state=00 duration=17 expire=45
timeout=0 flags=00000000 socktype=0 sockport=0 av_idx=0 use=3
origin-shaper=
reply-shaper=
```

```
per_ip_shaper=
class_id=0 ha_id=0 policy_dir=0 tunnel=/ vlan_cos=0/255
state=may_dirty
statistic(bytes/packets/allow_err): org=336/4/1 reply=336/4/1 tuples=2
tx speed(Bps/kbps): 19/0 rx speed(Bps/kbps): 19/0
orgin->sink: org pre->post, reply pre->post dev=89->137/137->89
gwy=10.10.200.3/22.1.1.22
hook=pre dir=org act=noop 22.1.1.22:25968->33.1.1.133:8(0.0.0.0:0)
hook=post dir=reply act=noop 33.1.1.133:25968->22.1.1.22:0(0.0.0.0:0)
src_mac=02:4c:a5:fc:6a:7f
misc=0 policy_id=1 pol_uuid_idx=566 auth_info=0 chk_client_info=0 vd=1:1
serial=000aa475 tos=ff/ff app_list=0 app=0 url_cat=0
sdwan_mbr_seq=0 sdwan_service_id=2
rpdb_link_id=ff000002 ngfwid=n/a
npu_state=0x5040001 no_offload
no_ofld_reason:  disabled-by-policy non-npu-intf
```

- User at 22.1.1.22 in VRF2 on spoke 1 pings 33.1.1.133 in VRF2 on spoke 2:

```
# diagnose sys session listsession info: proto=1 proto_state=00 duration=8 expire=53
timeout=0 flags=00000000 socktype=0 sockport=0 av_idx=0 use=3
origin-shaper=
reply-shaper=
per_ip_shaper=
class_id=0 ha_id=0 policy_dir=0 tunnel=/ vlan_cos=0/255
state=may_dirty
statistic(bytes/packets/allow_err): org=252/3/1 reply=252/3/1 tuples=2
tx speed(Bps/kbps): 30/0 rx speed(Bps/kbps): 30/0
orgin->sink: org pre->post, reply pre->post dev=113->137/137->113
gwy=10.10.200.3/22.1.1.22
hook=pre dir=org act=noop 22.1.1.22:28528->33.1.1.133:8(0.0.0.0:0)
hook=post dir=reply act=noop 33.1.1.133:28528->22.1.1.22:0(0.0.0.0:0)
src_mac=02:4c:a5:fc:6a:7f
misc=0 policy_id=1 pol_uuid_idx=566 auth_info=0 chk_client_info=0 vd=1:2
serial=000aa49f tos=ff/ff app_list=0 app=0 url_cat=0
sdwan_mbr_seq=0 sdwan_service_id=2
rpdb_link_id=ff000002 ngfwid=n/a
npu_state=0x5040001 no_offload
no_ofld_reason:  disabled-by-policy non-npu-intf
```

3. Check sessions on spoke 2:

   The output `vd=<vdom ID>:<VRF ID>` indicates that sessions are created in and stay in the corresponding VRFs.

   - User at 22.1.1.22 in VRF1 on spoke 1 pings 33.1.1.133 in VRF1 on spoke 2:

```
# diagnose sys session list
session info: proto=1 proto_state=00 duration=24 expire=38 timeout=0 flags=00000000
socktype=0 sockport=0 av_idx=0 use=3
origin-shaper=
reply-shaper=
per_ip_shaper=
class_id=0 ha_id=0 policy_dir=0 tunnel=/ vlan_cos=0/255
state=may_dirty npu
statistic(bytes/packets/allow_err): org=336/4/1 reply=336/4/1 tuples=2
tx speed(Bps/kbps): 13/0 rx speed(Bps/kbps): 13/0
orgin->sink: org pre->post, reply pre->post dev=138->92/92->138
gwy=33.1.1.133/10.10.200.2
```

```
hook=pre dir=org act=noop 22.1.1.22:25968->33.1.1.133:8(0.0.0.0:0)
hook=post dir=reply act=noop 33.1.1.133:25968->22.1.1.22:0(0.0.0.0:0)
misc=0 policy_id=1 pol_uuid_idx=630 auth_info=0 chk_client_info=0 vd=6:1
serial=000aa476 tos=ff/ff app_list=0 app=0 url_cat=0
rpdb_link_id=00000000 ngfwid=n/a
npu_state=0x4000001 no_offload
npu info: flag=0x00/0x82, offload=0/0, ips_offload=0/0, epid=0/0, ipid=0/0,
vlan=0x0000/0x0000
vlifid=0/0, vtag_in=0x0000/0x0000 in_npu=0/0, out_npu=0/0, fwd_en=0/0, qid=0/0
no_ofld_reason:  disabled-by-policy
```

- User at 22.1.1.22 in VRF2 on spoke 1 pings 33.1.1.133 in VRF2 on spoke2:

```
# diagnose sys session list
session info: proto=1 proto_state=00 duration=15 expire=46 timeout=0 flags=00000000
socktype=0 sockport=0 av_idx=0 use=3
origin-shaper=
reply-shaper=
per_ip_shaper=
class_id=0 ha_id=0 policy_dir=0 tunnel=/ vlan_cos=0/255
state=may_dirty npu
statistic(bytes/packets/allow_err): org=252/3/1 reply=252/3/1 tuples=2
tx speed(Bps/kbps): 16/0 rx speed(Bps/kbps): 16/0
orgin->sink: org pre->post, reply pre->post dev=138->115/115->138
gwy=33.1.1.133/10.10.200.2
hook=pre dir=org act=noop 22.1.1.22:28528->33.1.1.133:8(0.0.0.0:0)
hook=post dir=reply act=noop 33.1.1.133:28528->22.1.1.22:0(0.0.0.0:0)
misc=0 policy_id=1 pol_uuid_idx=630 auth_info=0 chk_client_info=0 vd=6:2
serial=000aa4a0 tos=ff/ff app_list=0 app=0 url_cat=0
rpdb_link_id=00000000 ngfwid=n/a
npu_state=0x4000001 no_offload
npu info: flag=0x00/0x82, offload=0/0, ips_offload=0/0, epid=0/0, ipid=0/0,
vlan=0x0000/0x0000
vlifid=0/0, vtag_in=0x0000/0x0000 in_npu=0/0, out_npu=0/0, fwd_en=0/0, qid=0/0
no_ofld_reason:  disabled-by-policy
```

Example 2

In this example, SLA health checks are sent from a spoke's VRF to the loopback on the hub that is in the same VRF.

To configure the health check:

```
config system sdwan
    config health-check
        edit "1"
            set server "11.11.22.11"
            set vrf 1
            set source 22.1.1.2
            set members 1 2
            config sla
                edit 1
                    set latency-threshold 200
                    set jitter-threshold 50
                next
            end
        next
```

```
    end
end
```

To check the health check status:

```
# diagnose sys sdwan health-check status  1
Health Check(1):
Seq(1 vd2-1): state(alive), packet-loss(0.000%) latency(0.023), jitter(0.002), mos(4.404),
bandwidth-up(0), bandwidth-dw(0), bandwidth-bi(0) sla_map=0x1
Seq(2 vd2-2): state(alive), packet-loss(0.000%) latency(0.022), jitter(0.002), mos(4.404),
bandwidth-up(0), bandwidth-dw(0), bandwidth-bi(0) sla_map=0x1
```

Example 3

In this example, when traffic from spoke 1 arrives at the hub on tunnel 1, it will egress the hub on tunnel 1 to go to other spokes. If traffic arrives on tunnel 2, it will egress on tunnel 2, and not tunnel 1.

To configure SD-WAN on the hub:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
            set service-sla-tie-break input-device
        next
    end
    config members
        edit 1
            set interface "p1"
        next
        edit 2
            set interface "p2"
        next
    end
    config health-check
        edit "1"
            set server "22.1.1.2"
            set members 1 2
            config sla
                edit 1
                next
            end
        next
    end
    config service
        edit 1
            set mode sla
            set dst "all"
            config sla
                edit "1"
                    set id 1
                next
            end
            set priority-members 1 2
```

```
            set tie-break input-device
        next
    end
end
```

To verify that traffic stays in the same overlay on ingress and egress on the hub:

1. Confirm that the SD-WAN service rule has `Tie break` set to `input-device` so that, when SLAs are met on all of the members, traffic prefers to egress on the same member as the input device:

```
# diagnose sys sdwan service

Service(1): Address Mode(IPV4) flags=0x200 use-shortcut-sla
 Tie break: input-device
  Gen(1), TOS(0x0/0x0), Protocol(0: 1->65535), Mode(sla), sla-compare-order
  Members(2):
    1: Seq_num(1 p1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0), selected
    2: Seq_num(2 p2), alive, sla(0x1), gid(0), cfg_order(1), local cost(0), selected
  Dst address(1):
        0.0.0.0-255.255.255.255
```

2. Use `diagnose sniffer packet` commands to verify that traffic ingress and egress are on the same overlay.

Multiple members per SD-WAN neighbor configuration

SD-WAN BGP neighbor configurations are used to define the SLA health check in which an SD-WAN member must meet to qualify as being up. When the SD-WAN member meets the SLA threshold, the FortiGate will apply the route map defined in the BGP neighbor's `route-map-out-preferable` option. If the SD-WAN member fails to meet the SLA, the FortiGate will apply the route map defined in the BGP neighbor's `route-map-out` option instead. This allows the FortiGate to advertise the health of the SD-WAN member to its BGP neighbor by advertising different community strings based on its SLA status.

For more information, refer to the following BGP examples in the FortiOS Administration Guide: Controlling traffic with BGP route mapping and service rules and Applying BGP route-map to multiple BGP neighbors.

In this enhancement, instead of selecting only one SD-WAN member per neighbor, multiple SD-WAN members can be selected. This allows the SD-WAN neighbor feature to support topologies where there are multiple SD-WAN overlays and/or underlays to a neighbor. The `minimum-sla-meet-members` option is used to configure the minimum number of members that must be in an SLA per neighbor for the preferable route map to be used.

```
config system sdwan
    config neighbor
        edit <ip>
            set member {<seq-num_1>} [<seq-num_2>] ... [<seq-num_n>]
            set minimum-sla-meet-members <integer>
        next
    end
end
```

`member {<seq-num_1>} [<seq-num_2>] ... [<seq-num_n>]`	Enter the member sequence number list. Multiple members can be defined.
`minimum-sla-meet-members <integer>`	Set the minimum number of members that meet SLA when the neighbor is preferred (1 - 255, default = 1). • If the number of in SLA members is less than the `minimum-sla-meet-members` value, the default route map will be used. • If the number of in SLA members is equal or larger than the `minimum-sla-meet-members` value, the preferable route map will be used.

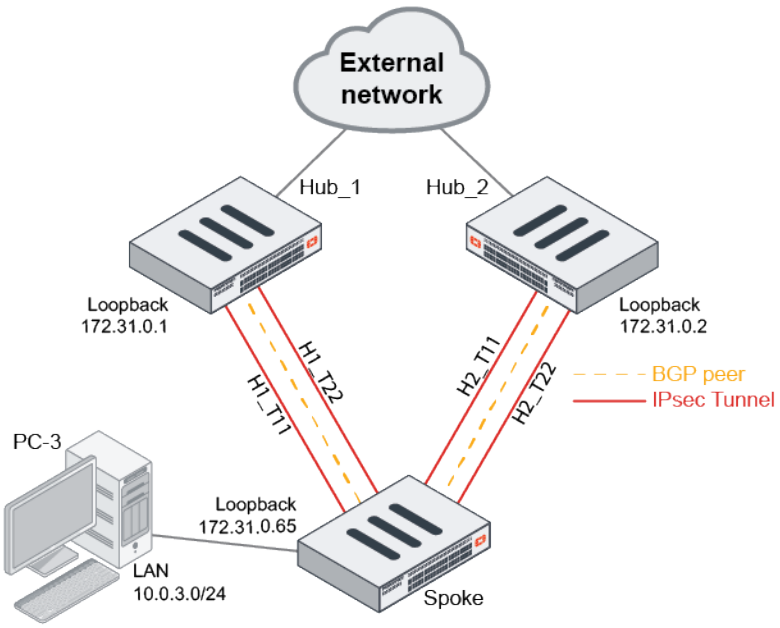
Example

In the following example, the spoke FortiGate has four tunnels: two tunnels to Hub_1 and two tunnels to Hub_2. The spoke has two BGP neighbors: one to Hub_1 and one to Hub-2. BGP neighbors are established on loopback IPs.

The SD-WAN neighbor plus `route-map-out-preferable`configuration is deployed on the spoke to achieve the following:

- If any tunnel to Hub_1 or Hub_2 is in SLA, the preferable route map will be applied on the BGP neighbor to Hub_1 or Hub_2.
- If both tunnels to Hub_1 or Hub_2 are out of SLA, the default route map will be applied on the BGP neighbor to Hub_1 or Hub_2.

The preferable route map and default route map are used to set different custom BGP communities as the spoke advertises its LAN routes to the hub. Each hub can translate communities into different BGP MED or AS prepends and signal them to the external peers to manipulate inbound traffic, thereby routing traffic to the spoke only when the SLAs are met on at least one of two VPN overlays. In this example, community string 10:1 signals to the neighbor that SLAs are met, and 10:2 signals that SLAs are not met.



To configure the BGP route maps and neighbors:

1. Configure an access list of prefixes to be matched:

```
config router access-list
    edit "net10"
        config rule
            edit 1
                set prefix 10.0.3.0 255.255.255.0
            next
        end
    next
end
```

2. Configure route maps for neighbors in SLA (preferable) and out of SLA (default):

```
config router route-map
    edit "in_sla"
        config rule
            edit 1
                set match-ip-address "net10"
                set set-community "10:1"
            next
        end
    next
    edit "out_sla"
        config rule
            edit 1
                set match-ip-address "net10"
                set set-community "10:2"
            next
        end
    next
end
```

3. Configure the BGP neighbors:

```
config router bgp
    set router-id 172.31.0.65
    config neighbor
        edit "172.31.0.1"
            set route-map-out "out_sla"
            set route-map-out-preferable "in_sla"
            set update-source "Loopback0"
        next
        edit "172.31.0.2"
            set route-map-out "out_sla"
            set route-map-out-preferable "in_sla"
            set update-source "Loopback0"
        next
    end
    config network
        edit 1
            set prefix 10.0.3.0 255.255.255.0
        next
    end
end
```

To configure SD-WAN:

1. Configure the SD-WAN members:

```
config system sdwan
    set status enable
    config members
        edit 1
            set interface "H1_T11"
            set source 172.31.0.65
        next
        edit 4
            set interface "H1_T22"
            set source 172.31.0.65
        next
        edit 6
            set interface "H2_T11"
            set source 172.31.0.65
        next
        edit 9
            set interface "H2_T22"
            set source 172.31.0.65
        next
    end
end
```

2. Configure the health check that must be met:

```
config system sdwan
    config health-check
        edit "HUB"
            set server "172.31.100.100"
            set members 0
            config sla
                edit 1
                    set link-cost-factor latency
                    set latency-threshold 100
                next
            end
        next
    end
end
```

3. Configure the SD-WAN neighbors:

```
config system sdwan
    config neighbor
        edit "172.31.0.1"
            set member 1 4
            set health-check "HUB"
            set sla-id 1
            set minimum-sla-meet-members 1
        next
        edit "172.31.0.2"
            set member 6 9
            set health-check "HUB"
            set sla-id 1
            set minimum-sla-meet-members 1
```

```
            next
        end
    end
```

To verify that when two members to Hub_1/Hub_2 are in SLA, the preferable route map is be applied on BGP neighbors to Hub_1/Hub_2:

```
Branch1_A_FGT (root) # diagnose sys sdwan health-check
Health Check(HUB):
Seq(1 H1_T11): state(alive), packet-loss(0.000%) latency(0.209), jitter(0.017), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999999), bandwidth-bi(1999998) sla_map=0x1
Seq(4 H1_T22): state(alive), packet-loss(0.000%) latency(0.171), jitter(0.004), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x1
Seq(6 H2_T11): state(alive), packet-loss(0.000%) latency(0.175), jitter(0.014), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999999), bandwidth-bi(1999998) sla_map=0x1
Seq(9 H2_T22): state(alive), packet-loss(0.000%) latency(0.176), jitter(0.019), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x1
```

```
# diagnose sys sdwan neighbor
Neighbor(172.31.0.1): member(1 4 )role(standalone)
        Health-check(HUB:1)  sla-pass selected alive
Neighbor(172.31.0.2): member(6 9 )role(standalone)
        Health-check(HUB:1)  sla-pass selected alive
```

On Hub_1 and Hub_2, the expected communities have been attached into the spoke's LAN route:

```
Hub_1_FGT (root) # get router info bgp network 10.0.3.0/24
VRF 0 BGP routing table entry for 10.0.3.0/24
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  Original VRF 0
  Local, (Received from a RR-client)
    172.31.0.65 from 172.31.0.65 (172.31.0.65)
      Origin IGP metric 0, localpref 100, valid, internal, best
      Community: 10:1
      Last update: Wed Dec 29 22:38:29 2021
```

```
Hub_2_FGT (root) # get router info bgp network 10.0.3.0/24
VRF 0 BGP routing table entry for 10.0.3.0/24
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  Original VRF 0
  Local, (Received from a RR-client)
    172.31.0.65 from 172.31.0.65 (172.31.0.65)
      Origin IGP metric 0, localpref 100, valid, internal, best
      Community: 10:1

      Last update: Wed Dec 29 22:43:10 2021
```

If one member for each neighbor becomes out of SLA, the preferable route map is still applied:

```
Branch1_A_FGT (root) # diagnose sys sdwan health-check
Health Check(HUB):
Seq(1 H1_T11): state(alive), packet-loss(0.000%) latency(120.207), jitter(0.018), mos
(4.338), bandwidth-up(999999), bandwidth-dw(999997), bandwidth-bi(1999996) sla_map=0x0
Seq(4 H1_T22): state(alive), packet-loss(0.000%) latency(0.182), jitter(0.008), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x1
Seq(6 H2_T11): state(alive), packet-loss(0.000%) latency(120.102), jitter(0.009), mos
```

```
(4.404), bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x0
Seq(9 H2_T22): state(alive), packet-loss(0.000%) latency(0.176), jitter(0.009), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999997), bandwidth-bi(1999996) sla_map=0x1

# diagnose sys sdwan neighbor
Neighbor(172.31.0.1): member(1 4 )role(standalone)
        Health-check(HUB:1)  sla-pass selected alive
Neighbor(172.31.0.2): member(6 9 )role(standalone)
        Health-check(HUB:1)  sla-pass selected alive

Hub_1_FGT (root) # get router info bgp network 10.0.3.0/24
VRF 0 BGP routing table entry for 10.0.3.0/24
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  Original VRF 0
  Local, (Received from a RR-client)
    172.31.0.65 from 172.31.0.65 (172.31.0.65)
      Origin IGP metric 0, localpref 100, valid, internal, best
      Community: 10:1
      Last update: Thu Dec 30 10:44:47 2021

Hub_2_FGT (root) # get router info bgp network 10.0.3.0/24
VRF 0 BGP routing table entry for 10.0.3.0/24
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  Original VRF 0
  Local, (Received from a RR-client)
    172.31.0.65 from 172.31.0.65 (172.31.0.65)
      Origin IGP metric 0, localpref 100, valid, internal, best
      Community: 10:1
      Last update: Wed Dec 29 22:43:10 2021
```

If both members for Hub_1 become out of SLA, the default route map is applied:

```
Branch1_A_FGT (root) # diagnose sys sdwan health-check
Health Check(HUB):
Seq(1 H1_T11): state(alive), packet-loss(0.000%) latency(120.194), jitter(0.018), mos
(4.338), bandwidth-up(999999), bandwidth-dw(999997), bandwidth-bi(1999996) sla_map=0x0
Seq(4 H1_T22): state(alive), packet-loss(0.000%) latency(120.167), jitter(0.006), mos
(4.338), bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x0
Seq(6 H2_T11): state(alive), packet-loss(0.000%) latency(120.180), jitter(0.012), mos
(4.338), bandwidth-up(999999), bandwidth-dw(999998), bandwidth-bi(1999997) sla_map=0x0
Seq(9 H2_T22): state(alive), packet-loss(0.000%) latency(0.170), jitter(0.005), mos(4.404),
bandwidth-up(999999), bandwidth-dw(999997), bandwidth-bi(1999996) sla_map=0x1

# diagnose sys sdwan  neighbor
Neighbor(172.31.0.1): member(1 4 )role(standalone)
        Health-check(HUB:1)  sla-fail alive
Neighbor(172.31.0.2): member(6 9 )role(standalone)
        Health-check(HUB:1)  sla-pass selected alive

Hub_1_FGT (root) # get router info bgp network 10.0.3.0/24
VRF 0 BGP routing table entry for 10.0.3.0/24
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  Original VRF 0
  Local, (Received from a RR-client)
    172.31.0.65 from 172.31.0.65 (172.31.0.65)
      Origin IGP metric 0, localpref 100, valid, internal, best
```

```
      Community: 10:2
      Last update: Thu Dec 30 10:57:33 2021

Hub_2_FGT (root) # get router info bgp network 10.0.3.0/24
VRF 0 BGP routing table entry for 10.0.3.0/24
Paths: (1 available, best #1, table Default-IP-Routing-Table)
  Not advertised to any peer
  Original VRF 0
  Local, (Received from a RR-client)
    172.31.0.65 from 172.31.0.65 (172.31.0.65)
      Origin IGP metric 0, localpref 100, valid, internal, best
      Community: 10:1
      Last update: Wed Dec 29 22:43:10 2021
```

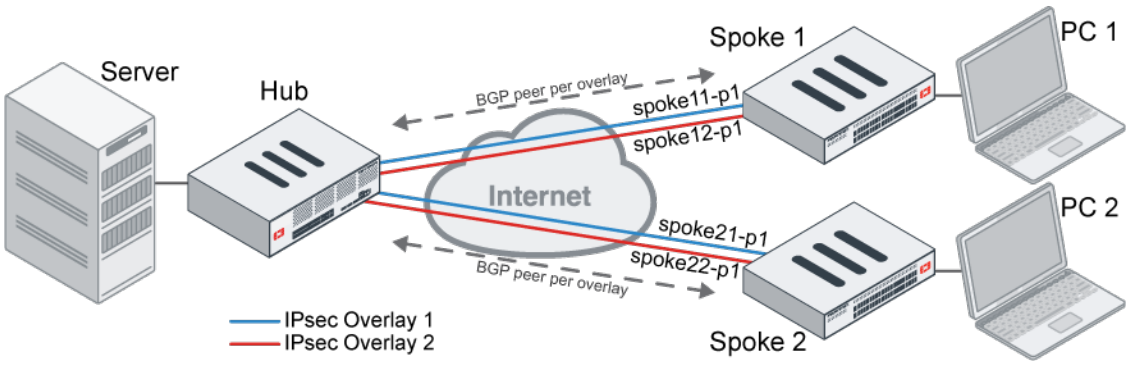
SD-WAN in large scale deployments

SD-WAN with ADVPN configurations in large-scale deployments is improved.

- Phase 2 selectors can be used to inject IKE routes on the ADVPN shortcut tunnel.

  When configuration method (`mode-cfg`) is enabled in IPsec phase 1 configuration, enabling `mode-cfg-allow-client-selector` allows custom phase 2 selectors to be configured. By also enabling the addition of a route to the peer destination selector (`add-route`) in the phase 1 configuration, IKE routes based on the phase 2 selectors can be injected. This means that routes do not need to be reflected on the hub to propagate them between spokes, avoiding possible BGP daemon process load issues and improving network scalability in a large-scale ADVPN network.
- Route map rules can apply priorities to BGP routes.

  On the hub, priorities can be set in a route map's rules, and the route map can be applied on BGP routes. This allows the hub to mark the preferred path learned from the spokes with a higher priority, instead of using multiple SD-WAN policy routes on the hub. When a preferred outbound route map (`route-map-out-preferable`) is also configured in an SD-WAN neighbor on the spoke, deploying SD-WAN rules on the hub to steer traffic from the hub to a spoke is unnecessary.
- SD-WAN members' local cost can be exchanged on the ADVPN shortcut tunnel so that spokes can use the remote cost as tiebreak to select a preferred shortcut. If multiple shortcuts originate from the same member to different members on the same remote spoke, then the remote cost on the shortcuts is used as the tiebreak to decide which shortcut is preferred.



In this example, SD-WAN is configured on an ADVPN network with a BGP neighbor per overlay.

Instead of reflecting BGP routes with the route-reflector on the hub, when the shortcuts are triggered, IKE routes on the shortcuts are directly injected based on the configured phase 2 selectors to allow routes to be exchanged between spokes.

Routes between the hub and the spokes are exchanged by BGP, and the spokes use the default route to send spoke-to-spoke traffic to the hub and trigger the shortcuts.

Instead of configuring SD-WAN rules on the hub, different priorities are configured on the BGP routes by matching different BGP communities to steer traffic from the hub to the spokes.

To configure Spoke 1:

1. Configure phase 1:

```
config vpn ipsec phase1-interface
    edit "spoke11-p1"
        ...
        set ike-version 2
        set net-device enable
        set add-route enable
        set mode-cfg enable
        set auto-discovery-receiver enable
        set mode-cfg-allow-client-selector enable
        ...
    next
    edit "spoke12-p1"
       ...
        set ike-version 2
        set net-device enable
        set add-route enable
        set mode-cfg enable
        set auto-discovery-receiver enable
        set mode-cfg-allow-client-selector enable
    next
end
```

2. Configure phase 2:

```
config vpn ipsec phase2-interface
    edit "spoke11-p2"
        ...
        set src-name "LAN_Net"
        set dst-name "all"
    next
    edit "spoke12-p2"
        ...
        set src-name "LAN_Net"
        set dst-name "all"
    next
end
```

3. Configure an address group:

   Spoke 1 uses LAN subnet 10.1-3.100.0/24.

```
config firewall addrgrp
    edit "LAN_Net"
        set member "10.1.100.0" "10.2.100.0" "10.3.100.0"
```

```
    next
end
```

4. Configure route maps:
   - If overlay 1 to the hub is in SLA, attach "65000:1" to the BGP routes advertised to the hub over overlay 1.
   - If overlay 2 to the hub is in SLA, attach "65000:2" to the BGP routes advertised to the hub over overlay 2.
   - If any overlay to the hub is out of SLA, attach "65000:9999" to the BGP routes advertised to the hub over any overlay.

```
config router route-map
    edit "HUB_CARRIER1"
        config rule
            edit 1
                set set-community "65000:1"
                ...
            next
        end
        ...
    next
    edit "HUB_CARRIER2"
        config rule
            edit 1
                set set-community "65000:2"
                ...
            next
        end
        ...
    next
    edit "HUB_BAD"
        config rule
            edit 1
                set set-community "65000:9999"
                ...
            next
        end
        ...
    next
end
```

5. Configure BGP and SD-WAN members and neighbors:

```
config router bgp
    set as 65412
    config neighbor
        edit "10.10.15.253"
            set remote-as 65412
            set route-map-out "HUB_BAD"
            set route-map-out-preferable "HUB_CARRIER1"
            ...
        next
        edit "10.10.16.253"
            set remote-as 65412
            set route-map-out "HUB_BAD"
            set route-map-out-preferable "HUB_CARRIER2"
            ...
        next
```

```
    end
end
config system sdwan
    config members
        edit 1
            set interface "spoke11-p1"
        next
        edit 2
            set interface "spoke12-p1"
        next
    end
    config neighbor
        edit "10.10.15.253"
            set member 1
            set health-check "1"
            set sla-id 1
        next
        edit "10.10.16.253"
            set member 2
            set health-check "11"
            set sla-id 1
        next
    end
end
```

To configure Spoke 2:

1. Configure phase 1:

```
config vpn ipsec phase1-interface
    edit "spoke21-p1"
        ...
        set ike-version 2
        set net-device enable
        set add-route enable
        set mode-cfg enable
        set auto-discovery-receiver enable
        set mode-cfg-allow-client-selector enable
        ...
    next
    edit "spoke22-p1"
       ...
        set ike-version 2
        set net-device enable
        set add-route enable
        set mode-cfg enable
        set auto-discovery-receiver enable
        set mode-cfg-allow-client-selector enable
    next
end
```

2. Configure phase 2:

```
config vpn ipsec phase2-interface
    edit "spoke21-p2"
        ...
```

```
            set src-name "LAN_Net"
            set dst-name "all"
        next
        edit "spoke22-p2"
            ...
            set src-name "LAN_Net"
            set dst-name "all"
        next
    end
```

3. Configure an address group:

   Spoke 2 uses LAN subnet 192.168.5-7.0/24.

```
config firewall addrgrp
    edit "LAN_Net"
        set member "192.168.5.0" "192.168.6.0" "192.168.7.0"
    next
end
```

4. Configure route maps:
   - If overlay 1 to the hub is in SLA, attach "65000:1" to the BGP routes advertised to the hub over overlay 1.
   - If overlay 2 to the hub is in SLA, attach "65000:2" to the BGP routes advertised to the hub over overlay 2.
   - If any overlay to the hub is out of SLA, attach "65000:9999" to the BGP routes advertised to the hub over any overlay.

```
config router route-map
    edit "HUB_CARRIER1"
        config rule
            edit 1
                set set-community "65000:1"
                ...
            next
        end
        ...
    next
    edit "HUB_CARRIER2"
        config rule
            edit 1
                set set-community "65000:2"
                ...
            next
        end
        ...
    next
    edit "HUB_BAD"
        config rule
            edit 1
                set set-community "65000:9999"
                ...
            next
        end
        ...
    next
end
```

5. Configure BGP and SD-WAN members and neighbors:

```
config router bgp
    set as 65412
    config neighbor
        edit "10.10.15.253"
            set remote-as 65412
            set route-map-out "HUB_BAD"
            set route-map-out-preferable "HUB_CARRIER1"
            ...
        next
        edit "10.10.16.253"
            set remote-as 65412
            set route-map-out "HUB_BAD"
            set route-map-out-preferable "HUB_CARRIER2"
            ...
        next
    end
end
```

```
config system sdwan
    config members
        edit 1
            set interface "spoke21-p1"
            set cost 100
        next
        edit 2
            set interface "spoke22-p1"
            set cost 200
        next
    end
    config neighbor
        edit "10.10.15.253"
            set member 1
            set health-check "1"
            set sla-id 1
        next
        edit "10.10.16.253"
            set member 2
            set health-check "11"
            set sla-id 1
        next
    end
end
```

To configure the hub:

1. Configure the route maps:
   - Set the priority to 100 for routes with community 65000:1, indicating that they are in SLA for overlay 1.
   - Set the priority to 200 for routes with community 65000:2, indicating that they are in SLA for overlay 2.
   - Set the priority to 9999 for routes with community 65000:9999, indicating that they are out of SLA for any overlay.

```
config router route-map
     edit "Set_Pri"
        config rule
            edit 1
                set match-community "comm_65000:1"
```

```
                set set-priority 100
            next
            edit 2
                set match-community "comm_65000:2"
                set set-priority 200
            next
            edit 3
                set match-community "comm_65000:9999"
                set set-priority 9999
            next
        end
    next
end
```

2. Configure BGP:

```
config router bgp
    set as 65412
    config neighbor-group
        edit "advpn"
            set remote-as 65412
            set route-map-in "Set_Pri"
            ...
        next
        edit "advpn2"
            set remote-as 65412
            set route-map-in "Set_Pri"
            ...
        next
    end
    config neighbor-range
        edit 1
            set prefix 10.10.15.0 255.255.255.0
            set neighbor-group "advpn"
        next
        edit 2
            set prefix 10.10.16.0 255.255.255.0
            set neighbor-group "advpn2"
        next
    end
end
```

To test the configuration:

1. Check the routing tables on the spokes:

   Spoke 1:

```
spoke-1 (root) # get router info routing-table all
B*      0.0.0.0/0 [200/0] via 10.10.15.253 (recursive is directly connected, spoke11-
p1), 00:01:17, [1/0]         // default route to hub
                          [200/0] via 10.10.16.253 (recursive is directly connected,
spoke12-p1), 00:01:17, [1/0]
B       9.0.0.0/24 [200/0] via 10.10.15.253 (recursive is directly connected, spoke11-
p1), 00:01:17, [1/0]        // route to the server behind hub
                           [200/0] via 10.10.16.253 (recursive is directly connected,
spoke12-p1), 00:01:17, [1/0]
```

```
C       10.1.100.0/24 is directly connected, port2            // route to PC 1
C       10.10.15.0/24 is directly connected, spoke11-p1       // overlay 1
C       10.10.15.1/32 is directly connected, spoke11-p1
C       10.10.16.0/24 is directly connected, spoke12-p1       // overlay 2
C       10.10.16.1/32 is directly connected, spoke12-p1
```

Spoke 2:

```
spoke-2 (root) # get router info routing-table all
B*      0.0.0.0/0 [200/0] via 10.10.15.253 (recursive is directly connected, spoke21-
p1), 00:46:14, [1/0]         // default route to hub
                          [200/0] via 10.10.16.253 (recursive is directly connected,
spoke22-p1), 00:46:14, [1/0]
B       9.0.0.0/24 [200/0] via 10.10.15.253 (recursive is directly connected, spoke21-
p1), 00:46:18, [1/0]        // route to the server behind hub
                           [200/0] via 10.10.16.253 (recursive is directly connected,
spoke22-p1), 00:46:18, [1/0]
C       10.10.15.0/24 is directly connected, spoke21-p1       // overlay 1
C       10.10.15.2/32 is directly connected, spoke21-p1
C       10.10.16.0/24 is directly connected, spoke22-p1       // overlay 2
C       10.10.16.2/32 is directly connected, spoke22-p1
C       192.168.5.0/24 is directly connected, port2           // route to PC 2
```

2. Send traffic from PC 1 to PC 2 and trigger the shortcut:

   The IKE routes on the shortcut are directly injected based on the phase 2 selectors, and spoke-to-spoke traffic then goes directly through the shortcut instead of going through the hub.

   Spoke 1:

```
spoke-1 (root) # get router info routing-table static
S       192.168.5.0/24 [15/0] via spoke11-p1_0 tunnel 172.16.200.4 vrf 0, [1/0]
S       192.168.6.0/24 [15/0] via spoke11-p1_0 tunnel 172.16.200.4 vrf 0, [1/0]
S       192.168.7.0/24 [15/0] via spoke11-p1_0 tunnel 172.16.200.4 vrf 0, [1/0]
```

```
spoke-1 (root) # diagnose sniffer packet any 'host 192.168.5.44' 4
interfaces=[any]
filters=[host 192.168.5.44]
1.446306 port2 in 10.1.100.22 -> 192.168.5.44: icmp: echo request
1.446327 spoke11-p1_0 out 10.1.100.22 -> 192.168.5.44: icmp: echo request
1.446521 spoke11-p1_0 in 192.168.5.44 -> 10.1.100.22: icmp: echo reply
1.446536 port2 out 192.168.5.44 -> 10.1.100.22: icmp: echo reply
```

   Spoke 2:

```
spoke-2 (root) # get router info routing-table static
S       10.1.100.0/24 [15/0] via spoke21-p1_0 tunnel 10.10.15.1 vrf 0, [1/0]
S       10.2.100.0/24 [15/0] via spoke21-p1_0 tunnel 10.10.15.1 vrf 0, [1/0]
S       10.3.100.0/24 [15/0] via spoke21-p1_0 tunnel 10.10.15.1 vrf 0, [1/0]
```

3. Confirm that the overlays are in SLA on the spokes:

   Spoke 1:

```
spoke-1 (root) # diagnose sys sdwan  neighbor
Neighbor(10.10.15.253): member(1)role(standalone)
        Health-check(1:1)  sla-pass selected alive
Neighbor(10.10.16.253): member(2)role(standalone)
        Health-check(11:1)  sla-pass selected alive
```

   Spoke 2:

```
spoke-2 (root) # diagnose sys sdwan neighbor
Neighbor(10.10.15.253): member(1)role(standalone)
        Health-check(1:1)  sla-pass selected alive
Neighbor(10.10.16.253): member(2)role(standalone)
        Health-check(11:1)  sla-pass selected alive
```

4. On the hub, check that the routes received from the spokes have the expected priorities:

```
hub (root) # diagnose ip route list | grep proto=11
tab=254 vf=0 scope=0 type=1 proto=11 prio=100 0.0.0.0/0.0.0.0/0->10.1.100.0/24
pref=0.0.0.0 gwy=10.10.15.1 dev=101(hub-phase1)
tab=254 vf=0 scope=0 type=1 proto=11 prio=200 0.0.0.0/0.0.0.0/0->10.1.100.0/24
pref=0.0.0.0 gwy=10.10.16.1 dev=102(hub2-phase1)
tab=254 vf=0 scope=0 type=1 proto=11 prio=100 0.0.0.0/0.0.0.0/0->192.168.5.0/24
pref=0.0.0.0 gwy=10.10.15.2 dev=101(hub-phase1)
tab=254 vf=0 scope=0 type=1 proto=11 prio=200 0.0.0.0/0.0.0.0/0->192.168.5.0/24
pref=0.0.0.0 gwy=10.10.16.2 dev=102(hub2-phase1)
```

The priority set by the hub's route map is based on the community string received from the spoke. The route with a lower priority value is selected, so traffic to Spoke 1 goes out on the hub-phase1 tunnel:

```
hub (root) # diagnose sniffer packet any 'host 9.0.0.2' 4
interfaces=[any]
filters=[host 9.0.0.2]
2.735456 R190 in 9.0.0.2 -> 10.1.100.22: icmp: echo request
2.735508 hub-phase1 out 9.0.0.2 -> 10.1.100.22: icmp: echo request
2.735813 hub-phase1 in 10.1.100.22 -> 9.0.0.2: icmp: echo reply
2.735854 R190 out 10.1.100.22 -> 9.0.0.2: icmp: echo reply
```

5. If overlay 1 goes out of SLA, the priorities of the routes on the hub are updated and traffic from the hub to Spoke 1 goes through overlay 2:

   Spoke 1:

```
spoke-1 (root) # diagnose sys sdwan  neighbor
Neighbor(10.10.15.253): member(1)role(standalone)
        Health-check(1:1)  sla-fail alive
Neighbor(10.10.16.253): member(2)role(standalone)
        Health-check(11:1)  sla-pass selected alive
```

   Spoke 2:

```
spoke-2 (root) # diagnose sys sdwan neighbor
Neighbor(10.10.15.253): member(1)role(standalone)
        Health-check(1:1)  sla-fail alive
Neighbor(10.10.16.253): member(2)role(standalone)
        Health-check(11:1)  sla-pass selected alive
```

   Hub:

```
hub (root) # diagnose ip route list | grep proto=11
tab=254 vf=0 scope=0 type=1 proto=11 prio=200 0.0.0.0/0.0.0.0/0->10.1.100.0/24
pref=0.0.0.0 gwy=10.10.16.1 dev=102(hub2-phase1)
tab=254 vf=0 scope=0 type=1 proto=11 prio=9999 0.0.0.0/0.0.0.0/0->10.1.100.0/24
pref=0.0.0.0 gwy=10.10.15.1 dev=101(hub-phase1)
tab=254 vf=0 scope=0 type=1 proto=11 prio=200 0.0.0.0/0.0.0.0/0->192.168.5.0/24
pref=0.0.0.0 gwy=10.10.16.2 dev=102(hub2-phase1)
tab=254 vf=0 scope=0 type=1 proto=11 prio=9999 0.0.0.0/0.0.0.0/0->192.168.5.0/24
pref=0.0.0.0 gwy=10.10.15.2 dev=101(hub-phase1)
```

```
hub (root) # diagnose sniffer packet any 'host 9.0.0.2' 4
interfaces=[any]
filters=[host 9.0.0.2]
3.550181 R190 in 9.0.0.2 -> 10.1.100.22: icmp: echo request
3.550234 hub2-phase1 out 9.0.0.2 -> 10.1.100.22: icmp: echo request
3.550713 hub2-phase1 in 10.1.100.22 -> 9.0.0.2: icmp: echo reply
3.550735 R190 out 10.1.100.22 -> 9.0.0.2: icmp: echo reply
```

6. Trigger shortcuts between Spoke 1 and Spoke 2:
   - Shortcuts spoke11-p1_1 and spoke11-p1_0 originate from spoke11-p1.
   - spoke11-p1_1 corresponds to spoke21-p1_0 on Spoke 2.
   - spoke11-p1_0 corresponds to spoke22-p1_0 on Spoke 2.

   Spoke 1:

```
spoke-1 (root) # diagnose sys sdwan service

Service(1): Address Mode(IPV4) flags=0x200 use-shortcut-sla
 Tie break: cfg
  Gen(12), TOS(0x0/0x0), Protocol(0: 1->65535), Mode(sla), sla-compare-number
  Service role: standalone
  Member sub interface(4):
    3: seq_num(1), interface(spoke11-p1):
       1: spoke11-p1_0(75)
       2: spoke11-p1_1(76)
  Members(4):
    1: Seq_num(1 spoke11-p1_1), alive, sla(0x1), gid(0), remote cost(100), cfg_order(0),
local cost(0), selected
    2: Seq_num(1 spoke11-p1_0), alive, sla(0x1), gid(0), remote cost(200), cfg_order(0),
local cost(0), selected
    3: Seq_num(1 spoke11-p1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    4: Seq_num(2 spoke12-p1), alive, sla(0x2), gid(0), cfg_order(1), local cost(0),
selected
  Src address(1):
        10.1.100.0-10.1.100.255

  Dst address(1):
        0.0.0.0-255.255.255.255
```

   Spoke 2:

```
spoke-2 (root) # diagnose sys sdwan service

Service(1): Address Mode(IPV4) flags=0x200 use-shortcut-sla
 Tie break: cfg
  Gen(9), TOS(0x0/0x0), Protocol(0: 1->65535), Mode(sla), sla-compare-number
  Service role: standalone
  Member sub interface(4):
    2: seq_num(1), interface(spoke21-p1):
       1: spoke21-p1_0(68)
    4: seq_num(2), interface(spoke22-p1):
       1: spoke22-p1_0(67)
  Members(4):
    1: Seq_num(1 spoke21-p1_0), alive, sla(0x1), gid(0), cfg_order(0), local cost(100),
selected
    2: Seq_num(1 spoke21-p1), alive, sla(0x1), gid(0), cfg_order(0), local cost(100),
```

```
selected
    3: Seq_num(2 spoke22-p1_0), alive, sla(0x2), gid(0), cfg_order(1), local cost(200),
selected
    4: Seq_num(2 spoke22-p1), alive, sla(0x2), gid(0), cfg_order(1), local cost(200),
selected
  Src address(1):
        192.168.5.0-192.168.5.255

  Dst address(1):
        0.0.0.0-255.255.255.255
```

7. On Spoke 2, increase the cost of spoke21-p1_0 to 300.

```
spoke-2 (root) # config system sdwan
    config members
        edit 1
            set interface "spoke21-p1"
            set cost 300
        next
    end
end
```

The new cost is learned by the spoke11-p1_1 shortcut on Spoke 1, and that shortcut is no longer preferred due to its higher remote cost:

Spoke 1:

```
spoke-1 (root) # diagnose sys sdwan service

Service(1): Address Mode(IPV4) flags=0x200 use-shortcut-sla
 Tie break: cfg
  Gen(13), TOS(0x0/0x0), Protocol(0: 1->65535), Mode(sla), sla-compare-number
  Service role: standalone
  Member sub interface(4):
    3: seq_num(1), interface(spoke11-p1):
       1: spoke11-p1_0(78)
       2: spoke11-p1_1(79)
  Members(4):
    1: Seq_num(1 spoke11-p1_0), alive, sla(0x1), gid(0), remote cost(200), cfg_order(0),
local cost(0), selected
    2: Seq_num(1 spoke11-p1_1), alive, sla(0x1), gid(0), remote cost(300), cfg_order(0),
local cost(0), selected
    3: Seq_num(1 spoke11-p1), alive, sla(0x1), gid(0), cfg_order(0), local cost(0),
selected
    4: Seq_num(2 spoke12-p1), alive, sla(0x2), gid(0), cfg_order(1), local cost(0),
selected
  Src address(1):
        10.1.100.0-10.1.100.255

  Dst address(1):
        0.0.0.0-255.255.255.255
```

Spoke 2:

```
spoke-2 (root) # diagnose sys sdwan service

Service(1): Address Mode(IPV4) flags=0x200 use-shortcut-sla
 Tie break: cfg
```

```
  Gen(1), TOS(0x0/0x0), Protocol(0: 1->65535), Mode(sla), sla-compare-number
  Service role: standalone
  Member sub interface(4):
    2: seq_num(2), interface(spoke22-p1):
       1: spoke22-p1_0(70)
    4: seq_num(1), interface(spoke21-p1):
       1: spoke21-p1_0(71)
  Members(4):
    1: Seq_num(2 spoke22-p1_0), alive, sla(0x2), gid(0), cfg_order(1), local cost(200),
selected
    2: Seq_num(2 spoke22-p1), alive, sla(0x2), gid(0), cfg_order(1), local cost(200),
selected
    3: Seq_num(1 spoke21-p1_0), alive, sla(0x1), gid(0), cfg_order(0), local cost(300),
selected
    4: Seq_num(1 spoke21-p1), alive, sla(0x1), gid(0), cfg_order(0), local cost(300),
selected
  Src address(1):
        192.168.5.0-192.168.5.255

  Dst address(1):
        0.0.0.0-255.255.255.255
```

Route map rules and BGP routes

Route map rules can apply priorities to BGP routes. On the hub, priorities can be set in a route map's rules, and the route map can be applied on BGP routes. This allows the hub to mark the preferred path learned from the spokes with a higher priority, instead of using multiple SD-WAN policy routes on the hub. When a preferred outbound route map (`route-map-out-preferable`) is also configured in an SD-WAN neighbor on the spoke, deploying SD-WAN rules on the hub to steer traffic from the hub to a spoke is unnecessary.

For details, see SD-WAN in large scale deployments on page 52.

BGP socket limit increase

The BGP socket limit has been increased to 80,000.

See also SD-WAN in large scale deployments on page 52.

IKE embryonic limit

Administrators can configure the maximum number of IPsec tunnels to simultaneously negotiate.

To configure the embryonic limit:

```
config system ike
    set embryonic-limit {integer}
end
```

See also SD-WAN in large scale deployments on page 52.

SD-WAN steering

7.2.0

Allow application category as an option for SD-WAN rule destination

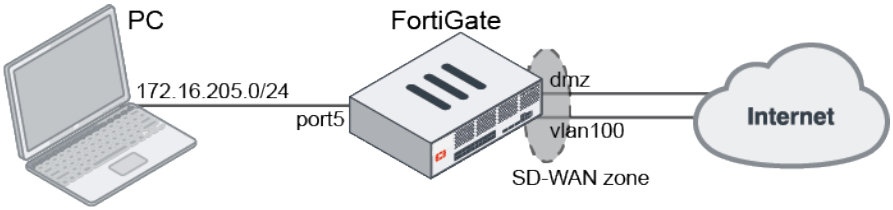
An application category can be selected as an SD-WAN service rule destination criterion. Previously, only application groups or individual applications could be selected.

```
config system sdwan
    config service
        edit <id>
            set internet-service enable
            set internet-service-app-ctrl-category <id_1> <id_2> ... <id_n>
        next
    end
end
```

To view the detected application categories details based on category ID, use `diagnose sys sdwan internet-service-app-ctrl-category-list <id>`.

Example

In this example, traffic steering is applied to traffic detected as video/audio (category ID 5) or email (category ID 21) and applies the lowest cost (SLA) strategy to this traffic. When costs are tied, the priority goes to member 1, dmz.



To configure application categories as an SD-WAN rule destination:

1. Configure the SD-WAN settings:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
```

```
        end
        config members
            edit 1
                set interface "dmz"
                set gateway 172.16.208.2
            next
            edit 2
                set interface "vlan100"
                set gateway 172.16.206.2
            next
        end
        config health-check
            edit "1"
                set server "8.8.8.8"
                set protocol dns
                set members 0
                config sla
                    edit 1
                    next
                end
            next
        end
    end
```

2. Configure the SD-WAN rule to use application categories 5 and 21:

```
config system sdwan
    config service
        edit 1
            set name "1"
            set mode sla
            set src "172.16.205.0"
            set internet-service enable
            set internet-service-app-ctrl-category 5 21
            config sla
                edit "1"
                    set id 1
                next
            end
            set priority-members 1 2
        next
    end
end
```

3. Configure the firewall policy:

```
config firewall policy
    edit 1
        set srcintf "port5"
        set dstintf "virtual-wan-link"
        set action accept
        set srcaddr 172.16.205.0
        set dstaddr "all"
        set schedule "always"
        set service "ALL"
        set utm-status enable
        set ssl-ssh-profile "certificate-inspection"
```

```
        set application-list "g-default"
    next
end
```

4. Verify that the traffic is sent over dmz:

```
# diagnose firewall proute list
list route policy info(vf=root):
id=2133590017(0x7f2c0001) vwl_service=1(1) vwl_mbr_seq=1 2 dscp_tag=0xff 0xff flags=0x0
tos=0x00 tos_mask=0x00 protocol=0 sport=0-65535 iif=0 dport=1-65535 path(2) oif=5(dmz)
oif=95(vlan100)
source(1): 172.16.205.0-172.16.205.255
destination wildcard(1): 0.0.0.0/0.0.0.0
internet service(2): (null)(0,5,0,0,0) (null)(0,21,0,0,0)
hit_count=469 last_used=2021-12-15 15:06:05
```

5. View some videos and emails on the PC, then verify the detected application details for each category:

```
# diagnose sys sdwan internet-service-app-ctrl-category-list 5
YouTube(31077 4294838537): 142.250.217.110 6 443 Wed Dec 15 15:39:50 2021
YouTube(31077 4294838537): 173.194.152.89 6 443 Wed Dec 15 15:37:20 2021
YouTube(31077 4294838537): 173.194.152.170 6 443 Wed Dec 15 15:37:37 2021
YouTube(31077 4294838537): 209.52.146.205 6 443 Wed Dec 15 15:37:19 2021

# diagnose sys sdwan internet-service-app-ctrl-category-list 21
Gmail(15817 4294836957): 172.217.14.197 6 443 Wed Dec 15 15:39:47 2021
```

6. Verify that the captured email traffic is sent over dmz:

```
# diagnose sniffer packet any 'host 172.217.14.197' 4
interfaces=[any]
filters=[host 172.217.14.197]
5.079814 dmz out 172.16.205.100.60592 -> 172.217.14.197.443: psh 2961561240 ack
2277134591
```

7. Edit the SD-WAN rule so that dmz has a higher cost and vlan100 is preferred.
8. Verify that the traffic is now sent over vlan100:

```
# diagnose firewall proute list
list route policy info(vf=root):
id=2134048769(0x7f330001) vwl_service=1(1) vwl_mbr_seq=2 1 dscp_tag=0xff 0xff flags=0x0
tos=0x00 tos_mask=0x00 protocol=0 sport=0-65535 iif=0 dport=1-65535 path(2) oif=95
(vlan100) oif=5(dmz)
source(1): 172.16.205.0-172.16.205.255
destination wildcard(1): 0.0.0.0/0.0.0.0
internet service(2): (null)(0,5,0,0,0) (null)(0,21,0,0,0)
hit_count=635 last_used=2021-12-15 15:55:43

# diagnose sniffer packet any 'host 172.217.14.197' 4
interfaces=[any]
filters=[host 172.217.14.197]
304.625168 vlan100 in 172.16.205.100.60592 -> 172.217.14.197.443: psh 2961572711 ack
2277139565
```

Add mean option score calculation and logging in performance SLA health checks

The mean opinion score (MOS) is a method of measuring voice quality using a formula that takes latency, jitter, packet loss, and the codec into account to produce a score from zero to five (0 - 5). The G.711, G.729, and G.722 codecs can be selected in the health check configurations, and an MOS threshold can be entered to indicate the minimum MOS score for the SLA to pass. The maximum MOS score will depend on which codec is used, since each codec has a theoretical maximum limit.

```
config system sdwan
    config health-check
          edit <name>
            set mos-codec {g711 | g729 | g722}
            config sla
                edit <id>
                    set link-cost-factor {latency jitter packet-loss mos}
                    set mos-threshold <value>
                next
            end
        next
    end
end
```

`mos-codec {g711 | g729 | g722}`	Set the VoIP codec to use for the MOS calculation (default = g711).
`link-cost-factor {latency jitter packet-loss mos}`	Set the criteria to base the link selection on.
`mos-threshold <value>`	Set the minimum MOS for the SLA to be marked as pass (1.0 - 5.0, default = 3.6).

Currently, the MOS cannot be used as the `link-cost-factor` to steer traffic in an SD-WAN rule.

To configure a health check to calculate the MOS:

```
config system sdwan
    set status enable
    config zone
        edit "virtual-wan-link"
        next
    end
    config members
        edit 1
            set interface "dmz"
            set gateway 172.16.208.2
        next
        edit 2
            set interface "port15"
            set gateway 172.16.209.2
        next
```

```
    end
    config health-check
        edit "Test_MOS"
            set server "2.2.2.2"
            set sla-fail-log-period 30
            set sla-pass-log-period 30
            set members 0
            set mos-codec g729
            config sla
                edit 1
                    set link-cost-factor mos
                    set mos-threshold "4.0"
                next
            end
        next
    end
end
```

To verify the MOS calculation results:

1. Verify the health check diagnostics:

```
# diagnose sys sdwan health-check
Health Check(Test_MOS):
Seq(1 dmz): state(alive), packet-loss(0.000%) latency(0.114), jitter(0.026), mos(4.123),
bandwidth-up(999999), bandwidth-dw(999997), bandwidth-bi(1999996) sla_map=0x1
Seq(2 port15): state(alive), packet-loss(0.000%) latency(0.100), jitter(0.008), mos
(4.123), bandwidth-up(999999), bandwidth-dw(999999), bandwidth-bi(1999998) sla_map=0x1

# diagnose sys sdwan sla-log Test_MOS 1
Timestamp: Tue Jan  4 11:23:06 2022, vdom root, health-check Test_MOS, interface: dmz,
status: up, latency: 0.151, jitter: 0.040, packet loss: 0.000%, mos: 4.123.
Timestamp: Tue Jan  4 11:23:07 2022, vdom root, health-check Test_MOS, interface: dmz,
status: up, latency: 0.149, jitter: 0.041, packet loss: 0.000%, mos: 4.123.

# diagnose sys sdwan sla-log Test_MOS 2
Timestamp: Tue Jan  4 11:25:09 2022, vdom root, health-check Test_MOS, interface:
port15, status: up, latency: 0.097, jitter: 0.009, packet loss: 0.000%, mos: 4.123.
Timestamp: Tue Jan  4 11:25:10 2022, vdom root, health-check Test_MOS, interface:
port15, status: up, latency: 0.097, jitter: 0.008, packet loss: 0.000%, mos: 4.123.
```

2. Change the `mos-codec` to `g722`. The diagnostics will now display different MOS values:

```
# diagnose sys sdwan health-check
Health Check(Test_MOS):
Seq(1 dmz): state(alive), packet-loss(0.000%) latency(0.150), jitter(0.031), mos(4.453),
bandwidth-up(999999), bandwidth-dw(999997), bandwidth-bi(1999996) sla_map=0x1
Seq(2 port15): state(alive), packet-loss(0.000%) latency(0.104), jitter(0.008), mos
(4.453), bandwidth-up(999999), bandwidth-dw(999999), bandwidth-bi(1999998) sla_map=0x1
```

3. Increase the latency on the link in port15. The calculated MOS value will decrease accordingly. In this example, port15 is out of SLA since its MOS value is now less than the 4.0 minimum:

```
# diagnose sys sdwan  health-check
Health Check(Test_MOS):
Seq(1 dmz): state(alive), packet-loss(0.000%) latency(0.106), jitter(0.022), mos(4.453),
bandwidth-up(999999), bandwidth-dw(999997), bandwidth-bi(1999996) sla_map=0x1
```

```
Seq(2 port15): state(alive), packet-loss(0.000%) latency(300.119), jitter(0.012), mos
(3.905), bandwidth-up(999999), bandwidth-dw(999999), bandwidth-bi(1999998) sla_map=0x0
```

Sample logs

```
date=2022-01-04 time=11:57:54 eventtime=1641326274876828300 tz="-0800" logid="0113022933"
type="event" subtype="sdwan" level="notice" vd="root" logdesc="SDWAN SLA notification"
eventtype="SLA" healthcheck="Test_MOS" slatargetid=1 interface="port15" status="up"
latency="300.118" jitter="0.013" packetloss="0.000" mos="3.905"
inbandwidthavailable="1000.00Mbps" outbandwidthavailable="1000.00Mbps"
bibandwidthavailable="2.00Gbps" inbandwidthused="0kbps" outbandwidthused="0kbps"
bibandwidthused="0kbps" slamap="0x0" metric="mos" msg="Health Check SLA status. SLA failed
due to being over the performance metric threshold."
```

```
date=2022-01-04 time=11:57:24 eventtime=1641326244286635920 tz="-0800" logid="0113022923"
type="event" subtype="sdwan" level="notice" vd="root" logdesc="SDWAN status"
eventtype="Health Check" healthcheck="Test_MOS" slatargetid=1 oldvalue="2" newvalue="1"
msg="Number of pass member changed."
```

```
date=2022-01-04 time=11:57:24 eventtime=1641326244286627260 tz="-0800" logid="0113022923"
type="event" subtype="sdwan" level="notice" vd="root" logdesc="SDWAN status"
eventtype="Health Check" healthcheck="Test_MOS" slatargetid=1 member="2" msg="Member status
changed. Member out-of-sla."
```

```
date=2022-01-04 time=11:57:02 eventtime=1641326222516756500 tz="-0800" logid="0113022925"
type="event" subtype="sdwan" level="information" vd="root" logdesc="SDWAN SLA information"
eventtype="SLA" healthcheck="Test_MOS" slatargetid=1 interface="port15" status="up"
latency="0.106" jitter="0.007" packetloss="0.000" mos="4.453"
inbandwidthavailable="1000.00Mbps" outbandwidthavailable="1000.00Mbps"
bibandwidthavailable="2.00Gbps" inbandwidthused="0kbps" outbandwidthused="0kbps"
bibandwidthused="0kbps" slamap="0x1" msg="Health Check SLA status."
```

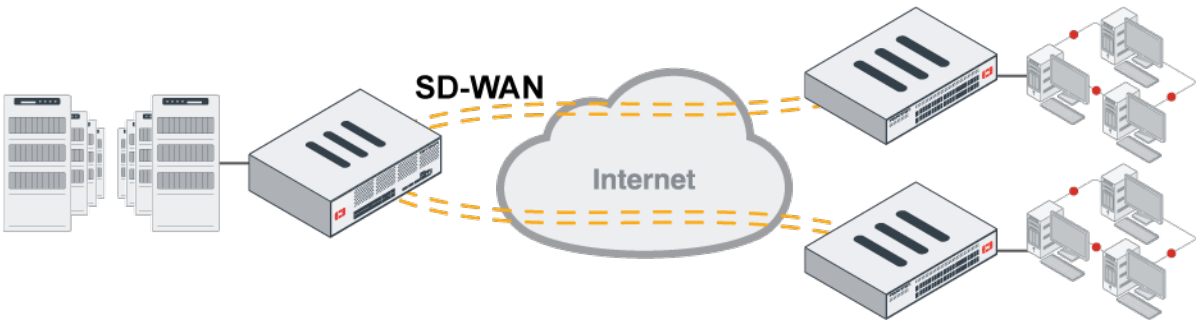
WAN remediation

7.2.0

Duplication on-demand when SLAs in the configured service are matched

SD-WAN packet duplication can be configured to be performed on-demand only when SLAs in the configured service are matched. When enabled, only the SLA health checks and targets that are used in the service rule are used to trigger the packet duplication.

```
config system sdwan
    config duplication
        edit 1
            set service-id 1
            set packet-duplication on-demand
            set sla-match-service {enable | disable}
        next
    end
end
```



In this example, two performance SLA health checks are configured, health1 and health2. The health1 SLA is used in an SD-WAN service rule called rule1. Packet duplication uses on-demand mode, so packets for duplication are matched based on rule1. It triggers duplication based on the status of the health checks.

Results are shown for various combinations of health check statuses when the SLA match service is enabled or disabled.

To configure SD-WAN:

```
config system sdwan
    set status enable
    set load-balance-mode usage-based
    config zone
        edit "virtual-wan-link"
```

```
            next
            edit "SASE"
            next
        end
        config members
            edit 1
                set interface "port5"
                set gateway 10.100.1.1
            next
            edit 2
                set interface "port4"
            next
        end
        config health-check
            edit "health1"
                set server "10.100.2.22"
                set members 0
                config sla
                    edit 1
                    next
                end
            next
            edit "health2"
                set server "10.100.2.23"
                set members 0
                config sla
                    edit 1
                    next
                end
            next
        end
        config service
            edit 1
                set name "rule1"
                set mode sla
                set dst "10.100.20.0"
                config sla
                    edit "health1"
                        set id 1
                    next
                end
                set priority-members 2 1
            next
        end
        config duplication
            edit 1
                set service-id 1
                set packet-duplication on-demand
                set sla-match-service enable
            next
        end
    end
```

Results

- When health1 (used in rule1) is out of SLA (`sla_map=0x0`) and health2 (not used) is in SLA (`sla_map=0x1`), the packet is duplicated (`dup=0x1(dup)`):

```
# diagnose sys sdwan health-check
Health Check(health1):
Seq(1 port5): state(alive), packet-loss(6.000%) latency(5.718), jitter(0.086), mos
(4.404), bandwidth-up(99995), bandwidth-dw(99995), bandwidth-bi(199990) sla_map=0x0
Seq(2 port4): state(alive), packet-loss(3.000%) latency(7.242), jitter(0.025), mos
(4.404), bandwidth-up(99998), bandwidth-dw(99999), bandwidth-bi(199997) sla_map=0x0
Health Check(health2):
Seq(1 port5): state(alive), packet-loss(0.000%) latency(0.700), jitter(0.075), mos
(4.404), bandwidth-up(99995), bandwidth-dw(99995), bandwidth-bi(199990) sla_map=0x1
Seq(2 port4): state(alive), packet-loss(0.000%) latency(0.244), jitter(0.021), mos
(4.404), bandwidth-up(99998), bandwidth-dw(99999), bandwidth-bi(199997) sla_map=0x1
```

```
# diagnose firewall proute list
id=2135031809(0x7f420001) vwl_service=1(rule1) vwl_mbr_seq=2 1 dscp_tag=0xfc 0xfc
flags=0x0 tos=0x00 tos_mask=0x00 protocol=0 sport=0-65535 iif=0 dport=1-65535 path(2)
oif=12(port4) measure=0x0(not measured) dup=0x1(dup) oif=13(port5) measure=0x0(not
measured) dup=0x1(dup)
destination(1): 10.100.20.0-10.100.20.255
source wildcard(1): 0.0.0.0/0.0.0.0
```

The sniffer output shows packets leaving from both interfaces in the zone:

```
# diagnose sniffer packet any "port 90" 4
interfaces=[any]
filters=[port 90]
2.403506 port2 in 172.16.205.11.59624 -> 10.100.20.33.90: syn 2098685816
2.403522 port5 out 10.100.1.250.59624 -> 10.100.20.33.90: syn 2098685816
2.403523 port4 out 10.100.1.250.59624 -> 10.100.20.33.90: syn 2098685816
```

```
# diagnose sys sdwan service

Service(1): Address Mode(IPV4) flags=0x200 use-shortcut-sla
 Tie break: cfg
  Gen(6), TOS(0x0/0x0), Protocol(0: 1->65535), Mode(sla), sla-compare-order
  Members(2):
    1: Seq_num(2 port4), alive, sla(0x0), gid(0), cfg_order(0), cost(0), selected
    2: Seq_num(1 port5), alive, sla(0x0), gid(0), cfg_order(1), cost(0), selected
  Dst address(1):
        10.100.20.0-10.100.20.255
```

- When health1 (used in rule1) is in SLA (`sla_map=0x1`) and health2 (not used) is out of SLA (`sla_map=0x0`), the packet is not duplicated (`dup=0x0(not dup)`):

```
# diagnose sys sdwan health-check
Health Check(health1):
Seq(1 port5): state(alive), packet-loss(0.000%) latency(0.684), jitter(0.064), mos
(4.404), bandwidth-up(99995), bandwidth-dw(99995), bandwidth-bi(199990) sla_map=0x1
Seq(2 port4): state(alive), packet-loss(0.000%) latency(0.222), jitter(0.015), mos
(4.404), bandwidth-up(99998), bandwidth-dw(99999), bandwidth-bi(199997) sla_map=0x1
Health Check(health2):
Seq(1 port5): state(alive), packet-loss(6.000%) latency(2.911), jitter(2.328), mos
(1.787), bandwidth-up(99995), bandwidth-dw(99996), bandwidth-bi(199990) sla_map=0x0
```

```
Seq(2 port4): state(alive), packet-loss(6.000%) latency(2.566), jitter(2.307), mos
(1.786), bandwidth-up(99998), bandwidth-dw(99999), bandwidth-bi(199997) sla_map=0x0

# diagnose firewall proute list
id=2135031809(0x7f420001) vwl_service=1(rule1) vwl_mbr_seq=2 1 dscp_tag=0xfc 0xfc
flags=0x0 tos=0x00 tos_mask=0x00 protocol=0 sport=0-65535 iif=0 dport=1-65535 path(2)
oif=12(port4) measure=0x0(not measured) dup=0x0(not dup) oif=13(port5) measure=0x0(not
measured) dup=0x0(not dup)
destination(1): 10.100.20.0-10.100.20.255
source wildcard(1): 0.0.0.0/0.0.0.0
```

The sniffer output shows packets leaving from only one interface:

```
# diagnose sniffer packet any "port 90" 4
interfaces=[any]
filters=[port 90]
3.330376 port2 in 172.16.205.11.38318 -> 10.100.21.33.90: syn 381919014
3.330395 port5 out 10.100.1.2.38318 -> 10.100.21.33.90: syn 381919014
4.327851 port2 in 172.16.205.11.38318 -> 10.100.21.33.90: syn 381919014
4.327855 port5 out 10.100.1.2.38318 -> 10.100.21.33.90: syn 381919014

# diagnose sys sdwan service

Service(1): Address Mode(IPV4) flags=0x200 use-shortcut-sla
 Tie break: cfg
  Gen(4), TOS(0x0/0x0), Protocol(0: 1->65535), Mode(sla), sla-compare-order
  Members(2):
    1: Seq_num(2 port4), alive, sla(0x1), gid(0), cfg_order(0), cost(0), selected
    2: Seq_num(1 port5), alive, sla(0x1), gid(0), cfg_order(1), cost(0), selected
  Dst address(1):
        10.100.20.0-10.100.20.255
```

- When the SLA match service is disabled, packets are only duplicated with all of the health checks are out of SLA:

```
config system sdwan
    config duplication
        edit 1
            set service-id 1
            set packet-duplication on-demand
            set sla-match-service disable
        next
    end
end
```

  - When health1 is out of SLA (`sla_map=0x0`) and health2 is in SLA (`sla_map=0x1`), the packet is not duplicated (`dup=0x0(not dup)`):

```
# diagnose sys sdwan health-check
Health Check(health1):
Seq(1 port5): state(alive), packet-loss(5.000%) latency(6.587), jitter(0.096), mos
(4.404), bandwidth-up(99995), bandwidth-dw(99995), bandwidth-bi(199990) sla_map=0x0
Seq(2 port4): state(alive), packet-loss(3.000%) latency(3.365), jitter(0.085), mos
(4.404), bandwidth-up(99998), bandwidth-dw(99999), bandwidth-bi(199997) sla_map=0x0
Health Check(health2):
Seq(1 port5): state(alive), packet-loss(0.000%) latency(0.837), jitter(0.192), mos
(4.404), bandwidth-up(99995), bandwidth-dw(99995), bandwidth-bi(199990) sla_map=0x1
Seq(2 port4): state(alive), packet-loss(0.000%) latency(0.330), jitter(0.081), mos
(4.404), bandwidth-up(99998), bandwidth-dw(99999), bandwidth-bi(199997) sla_map=0x1
```

```
# diagnose firewall proute list
list route policy info(vf=root):

id=2135097345(0x7f430001) vwl_service=1(rule1) vwl_mbr_seq=2 1 dscp_tag=0xfc 0xfc
flags=0x0 tos=0x00 tos_mask=0x00 protocol=0 sport=0-65535 iif=0 dport=1-65535 path(2)
oif=12(port4) measure=0x0(not measured) dup=0x0(not dup) oif=13(port5) measure=0x0
(not measured) dup=0x0(not dup)
destination(1): 10.100.20.0-10.100.20.255
source wildcard(1): 0.0.0.0/0.0.0.0
```

```
# diagnose sys sdwan service

Service(1): Address Mode(IPV4) flags=0x200 use-shortcut-sla
 Tie break: cfg
  Gen(1), TOS(0x0/0x0), Protocol(0: 1->65535), Mode(sla), sla-compare-order
  Members(2):
    1: Seq_num(2 port4), alive, sla(0x1), gid(0), cfg_order(0), cost(0), selected
    2: Seq_num(1 port5), alive, sla(0x1), gid(0), cfg_order(1), cost(0), selected
  Dst address(1):
        10.100.20.0-10.100.20.255
```

- When both health1 and health2 are out of SLA (`sla_map=0x0`), the packet is duplicated (`dup=0x1(dup)`).

If there are multiple targets in a performance SLA health check, and only one of the targets is used in the service that is defined in the duplication rule, and the SLA match service is disabled, then only that target triggers packet duplication. It is note required for all of the targets in the health check to miss SLA.

www.fortinet.com